IMPROVED SOLUTION TECHNIQUES FOR THE EIGENSTRUCTURE OF FRACTIONAL ORDER SYSTEMS

THESIS

Michele Lynn Devereaux
Captain, USAF

AFIT/GAE/AA/88D-08

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GAE/AA/88D-08

IMPROVED SOLUTION TECHNIQUES FOR THE EIGENSTRUCTURE OF FRACTIONAL ORDER SYSTEMS

THESIS

Michele Lynn Devereaux
Captain, USAF

AFIT/GAE/AA/88D-08

Approved for public release; distribution unlimited

AFIT/GAE/AA/88D-08

IMPROVED SOLUTION TECHNIQUES FOR THE EIGENSTRUCTURE OF FRACTIONAL ORDER SYSTEMS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Aeronautical Engineering

Michele Lynn Devereaux, B.S.

Captain, USAF

December, 1988

Acknowledgments

A wish to express a sincere "Thank You" to my thesis advisor, Lieutenant Colonel Ronald Bagley, without whom this work would not have been possible. His patience and support, as well as his faith in me, were limitless.

I would like to take this opportunity to express my deepest gratitude to my high school math instructor, Mr. Paul Tung. He cultivated my aptitude for mathematics, and instilled in me a deep appreciation for mathematics.

A special debt of gratitude is owed to my new husband, Pierre, for his unfailing support (and also for agreeing to postpone the honeymoon until after this thesis was finished).

Michele Lynn Devereaux

Table of Contents

List of Figures

List of Tables

List of Symbols

[ ]	square matrix
$[\ ]^T$	transpose of matrix
$[\ ]^{-1}$	inverse of matrix
$\{\ \}$	column vector
A	cross-sectional area
b_m	parameters of viscoelastic model
$[D(\lambda)]$	dynamical matrix
$D^\alpha[\ ]$	generalized derivative of order α
E	Young's modulus
E_n	parameters of viscoelastic model
$F[\ ]$	Fourier transform operator
$\{F(s)\}$	Laplace transform of the vector of forcing functions
i	square root of negative one
$[K(s)]$	viscoelastic stiffness matrix
$[\widetilde{K}]$	pseudo stiffness matrix of expanded equations of motion
$L[\ ]$	Laplace transform operator
L	length of a rod element
$[M]$	mass matrix
$[\widetilde{M}]$	pseudomass matrix of expanded equations of motion
s	Laplace parameter
$\{x(t)\}$	column vector of structural displacements
$\{X(s)\}$	Laplace transform of $\{x(t)\}$

α_n	paramters of viscoelastic model
β_m	paramters of viscoelastic model
$\epsilon(t)$	strain history
$\Gamma(\alpha)$	gamma function of α
λ	eigenvalue associated with expanded equations of motion
$\{\phi\}$	mode shape
μ	shift factor
$\sigma(t)$	stress history
ω	Fourier parameter and frequency

Abstract

The structural problem of a viscoelastically damped rod is considered. A four parameter fractional derivative viscoelastic model rather than the traditional viscous model is used to describe the stress-strain relationship. The introduction of fractional order derivatives leads to high order matrix equations, which are cumbersome and time consuming to solve. Thus, there exists a motivation to seek alternate solution techniques. An existing technique, modified matrix iteration, is presented, and a new one, employing spectrum shift concepts, is proposed. The spectrum shift technique is shown to be significantly more efficient.

IMPROVED SOLUTION TECHNIQUES FOR THE EIGENSTRUCTURE OF FRACTIONAL ORDER SYSTEMS

I. Introduction

The fractional derivative viscoelastic model has its earliest roots in Nutting's observations that fractional powers of time could model the stress relaxation phenomenon [5]. Gemant later noted that stiffness and damping properties of viscoelastic materials seemed porportional to fractional powers of frequency, implying that fractional order time differentials might be used to model the behavior [13]. Scott-Blair combined the ideas of Nutting and Gemant by proposing the use of fractional order time derivatives [2]. Caputo applied the concept to the viscoelasic behavior of geological strata [4]. Then he and Minardi showed that constitutive relationships employing the fractional calculus described the mechanical properties of some metals and glasses [5]. Bagley proposed incorporating fractional derivatives into finite element models of viscoelastically damped structures. Since then, he and Torvik have jointly published several papers demonstrating the feasibility and benefits of using fractional calculus. Of particular note is "A Theoretical Basis for the Application of Fractonal Calculus to Viscoelasticity" [5], which uses molecular theory to derive the existence of generalized derivatives. Their efforts have shown that fractional calculus is an attractive approach to modelling viscoelastically damped structures. The resulting model requires very few parameters and is often accurate over six decades of frequency [2].

Generalized calculus is not a new concept -- mathematicians have dealt with it for some time [9:115-118]. A generalized derivative is represented in this paper as

$$D^{\alpha}[x(t)].$$

The generalized derivative can be defined for complex α, but only real values will be considered here. Fractional derivatives are generalized derivatives with rational α. The term "fractional calculus" implies the use of fractional derivatives.

This thesis reviews the properties of generalized derivatives and the expanded equations of motion for a fractional order system describing a viscoelastically damped rod. The technique proposed by Bagley to solve for the eigenstructure is presented. A more efficient method is presented in Chapter V, along with some examples.

II. Brief Overview of Generalized Derivatives as Applied to Viscoelastic Materials

Before applying generalized derivatives to structural problems, it is necessary to understand the properties of generalized derivatives and their use in viscoelastic theory. As will be shown, generalized derivatives behave in much the same way as conventional derivatives. When used to model viscoelastic materials, generalized derivatives typically provide an excellent model over a broad range of frequencies [4]. To show how generalized derivatives can be used to model viscoelastic materials, it is appropriate to first present the properties of generalized derivatives, especially the Laplace and Fourier transforms. The generalized derivative is defined as [1:2]

$$D^{\alpha}[x(t)] \equiv \frac{1}{\Gamma(1-\alpha)} \frac{d}{dt} \int_0^t \frac{x(\tau)}{(t-\tau)^{\alpha}} \, d\tau \qquad \text{for } 0 \leq \alpha < 1 \tag{1}$$

Note that this definition is only valid for $\alpha < 1$. However, the definition requires only a slight modification for a generalized derivative of order greater than one. Let m be a nonnegative integer, and α defined as before. Then [1:11]

$$D^{m+\alpha}[x(t)] \equiv \frac{1}{\Gamma(1-\alpha)} \frac{d^{m+1}}{dt^{m+1}} \int_0^t \frac{x(\tau)}{(t-\tau)^{\alpha}} \, d\tau \qquad \text{for } 0 \leq \alpha < 1 \tag{2}$$

Although imposing in the time domain, in the Laplace (or Fourier) domain, the generalized derivative manifests itself as a fractional power of s (or ω). To calculate the Laplace transform, let $\tau = t - \eta$. Then,

$$D^{\alpha}[x(t)] = \frac{1}{\Gamma(1-\alpha)} \frac{d}{dt} \int_0^t \frac{x(t-\eta)}{\eta^{\alpha}} \, d\eta \qquad \text{for } 0 \leq \alpha < 1 \tag{3}$$

Applying Leibnitz's rule,

$$D^{\alpha}[x(t)] = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{1}{\eta^{\alpha}} \frac{\partial}{\partial t} x(t-\eta) \, d\eta + \frac{x(0)}{\Gamma(1-\alpha) t^{\alpha}} \qquad \text{for } 0 \leq \alpha < 1 \tag{4}$$

Noting that the integral is a time convolution, and that

$$L\left[\frac{t^{-\alpha}}{\Gamma(1-\alpha)}\right] = \frac{1}{s^{1-\alpha}} \tag{5}$$

the Laplace transform is

$$L[D^\alpha[x(t)]] = \frac{1}{s^{1-\alpha}}\left(sL[x(t)] - x(0)\right) + \frac{x(0)}{s^{1-\alpha}} \tag{6}$$

or, more simply,

$$L[D^\alpha[x(t)]] = s^\alpha L[x(t)] \tag{7}$$

where

$$L[x(t)] = \int_0^\infty x(t)\, e^{-st}\, dt \tag{8}$$

Notice that for initial conditions equal to zero, the Laplace transform of a generalized derivative of order α has the same property as the conventional derivative: the transform is s^α times the transform of the function. In fact, the generalized derivative satisfies many of the same properties as the conventional derivative, particularly linearity and the compostion property [1:8-10]

$$D^\alpha[y(t) + x(t)] = D^\alpha[y(t)] + D^\alpha[x(t)] \tag{9}$$

$$D^\alpha[D^\beta[x(t)] = D^{\alpha+\beta}[x(t)] \tag{10}$$

The Fourier transform is defined as

$$F[x(t)] \equiv \int_{-\infty}^\infty x(t)\, e^{-i\omega t}\, dt \tag{11}$$

If $x(t) = 0$ for $t < 0$, then the Fourier transform can be written as

$$F[x(t)] = \int_0^\infty x(t)\, e^{-i\omega t}\, dt \tag{12}$$

It is easily seen that the Fourier transform of a generalized derivative is

$$F[D^\alpha[x(t)]] = (i\omega)^\alpha F[x(t)] \tag{13}$$

In the preceding discussion, the only restriction placed on α was that it be a nonnegative real number less than one. However, for engineering applications, an irrational number can be approximated by a rational number. So α will now be restricted to be rational as well. Using the term "fractional derivative" will indicate this additional restriction.

To illustrate the use of fractional derivatives in viscoelastic theory, consider the standard linear viscoelastic model relating stress and strain [2]

$$\sigma(t) + \sum_{m=1}^{M} b_m \frac{d^m \sigma(t)}{dt^m} = E_0 \epsilon(t) + \sum_{n=1}^{N} E_n \frac{d^n \epsilon(t)}{dt^n} \tag{14}$$

Recalling Scott-Blair's proposal, replace the conventional derivatives by derivatives of fractional order. The result is the general form of the fractional derivative viscoelastic model [2]

$$labelsum \sigma(t) + \sum_{m=1}^{M} b_m D^{\beta_m}[\sigma(t)] = E_0 \epsilon(t) + \sum_{n=1}^{N} E_n D^{\alpha_n}[\epsilon(t)] \tag{15}$$

A large number of materials can be modelled by replacing each sum in Equation refsum by a single term involving a fractional derivative

$$\sigma(t) + b\, D^{\beta}[\sigma(t)] = E_0 \epsilon(t) + E_1 D^{\alpha}[\epsilon(t)] \tag{16}$$

Invoking the Second Law of Thermodynamics requires that [3]

$$\begin{array}{rclcrcl} E_0 & \geq & 0 & \quad & E_1 & \geq & bE_0 \\ E_1 & \geq & 0 & & \alpha & = & \beta \\ b & > & 0 & & & & \end{array} \tag{17}$$

These constraints ensure nonnegative energy dissipation and nonnegative work. The stress-strain relation in the Laplace domain is

$$\frac{\sigma(s)}{\epsilon(s)} = \frac{E_0 + E_1 s^{\alpha}}{1 + b s^{\alpha}} \tag{18}$$

This is known as the four parameter model, and has been shown to be very accurate over several decades of frequency [4, 13, 14].

III. Expanded Equations

Although the fractional derivative viscoelastic model may provide an excellent description of a material's properties, in order for it to be useful, its application to a structure must lead to a solvable problem. This chaper illustrates the existence of a solution by examining the finite element model of a viscoelastically damped rod. The equations of motion are developed using the elastic-viscoelastic correspondence principle, which states that a viscoelastic problem is equivalent to an elastic problem with the elastic moduli replaced by the appropriate viscoelastic moduli [7:42]. This chapter develops the finite element model of a viscoelastically damped rod, constrained at each end. Figure 1 shows a five degree-of- freedom rod, constrained at each end, with viscoelastic damping pads at each node. Assume the rod is uniform and purely elastic. Using standard finite element techniques, the stiffness matrix for the elastic rod is of the form [8:300]

$$[K_E] = \frac{EA}{L}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \tag{19}$$

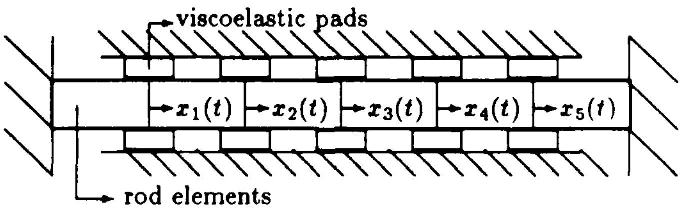

Figure 1. Finite Elements of Rod

where E is the Young's modulus for the material in the rod, A is the cross-sectional area, and L is the length of one element. Assume the modulus of the viscoelastic material is

$$E(s) = \frac{\sigma(s)}{\epsilon(s)} = \frac{E_0 + E_1 s^\alpha}{1 + bs^\alpha} \tag{20}$$

as derived in the previous chapter. The damping pads provide an out of phase shear stress to the rod. The shear stress is partially elastic and partially viscous, due to the real and imaginary parts of the modulus. As an example, let $\alpha = 1/2$, $b = 0$, and $s = i\omega$, where ω is an observed frequency of the system. Then

$$\begin{aligned} E(\omega) &= E_0 + E_1(i\omega)^{1/2} \\ &= E_0 + (\omega)^{1/2} E_1 e^{i\pi/4} \\ &= (E_0 + (\omega)^{1/2} E_1 \cos\tfrac{\pi}{4}) + (\omega)^{1/2} E_1 \sin\tfrac{\pi}{4} \end{aligned} \tag{21}$$

The real part represents the elastic component of the shear stress, and the imaginary part represents the viscous component, which is ninety degrees out of phase.

The contribution to the structure's stiffness matrix due to the viscoelastic pads is

$$G(s)[K_V] = \frac{G_0 + G_1 s^\alpha}{1 + bs^\alpha} \begin{bmatrix} A_1/t_1 & 0 & 0 & 0 & 0 \\ 0 & A_2/t_2 & 0 & 0 & 0 \\ 0 & 0 & A_3/t_3 & 0 & 0 \\ 0 & 0 & 0 & A_4/t_4 & 0 \\ 0 & 0 & 0 & 0 & A_5/t_5 \end{bmatrix} \tag{22}$$

where A_i is the area of the pad attached to the rod at i^{th} degree of freedom and t_i is the pad's thickness. The ratios A_i/t_i are the stiffness coefficients for the damping material at the corresponding degree of freedom. Then the stiffness matrix for the total structure is

$$[K(s)] = [K_E] + \frac{G_0 + G_1 s^\alpha}{1 + bs^\alpha}[K_V] \tag{23}$$

The mass matrix for the rod is [8:301-302]

$$[M] = \frac{\rho A L}{6} \begin{bmatrix} 4 & 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 & 4 \end{bmatrix} \tag{24}$$

where ρ is the density of the rod, and A and L are defined as above.

The equations of motion in the Laplace domain are

$$\left[s^2[M] + [K(s)]\right]\{X(s)\} = \{F(s)\} \tag{25}$$

where $\{F(s)\}$ is the Laplace transform of the forcing function. Setting $\{F(s)\} = 0$ yields the homogeneous equation, form which the eigenstructure is found.

To clear the denominator in $[K(s)]$, multiply through by $(1 + bs^{\alpha})$. Defining

$$[A_0] = G_0[K_V] + [K_E] \tag{26}$$

$$[A_q] = G_1[K_V] + b[K_E] \tag{27}$$

and expressing α as a ratio in lowest terms, q/m, gives

$$\left(s^{(2m+q)/m}\, b[M] + s^{2m/m}[M] + s^{q/m}[A_q] + [A_0]\right)\{X(s)\} = (1 + bs^{q/m})\{F(s)\} \tag{28}$$

In order to obtain an orthogonal transformation and decouple the equations of motion, cast the equations of motion in the following format

$$s^{1/m}[\widetilde{M}]\{\widetilde{X}(s)\} + [\widetilde{K}]\{\widetilde{X}(s)\} = \{\widetilde{F}(s)\} \tag{29}$$

$$[\widetilde{M}] = \begin{vmatrix} [0] & [0] & \cdots & [0] & b[M] \\ [0] & [0] & \cdots & b[M] & \vdots \\ \vdots & \vdots & \vdots & \vdots & [A_q] \\ [0] & b[M] & \cdots & [A_q] & \vdots \\ b[M] & \cdots & [A_q] & \cdots & [0] \end{vmatrix}$$

$$[\widetilde{K}] = \begin{vmatrix} [0] & [0] & \cdots & [0] & -b[M] & [0] \\ [0] & [0] & \cdots & -b[M] & \cdots & [0] \\ \vdots & \vdots & \vdots & \vdots & [-A_q] & \vdots \\ [0] & -b[M] & \cdots & [-A_q] & \cdots & [0] \\ -b[M] & \cdots & [-A_q] & \cdots & [0] & [0] \\ [0] & [0] & \cdots & [0] & [0] & [A_0] \end{vmatrix}$$

$$\{\widetilde{X}(s)\} = \begin{Bmatrix} s^{(2m-1)/m} & \{X(s)\} \\ s^{(2m-2)/m} & \{X(s)\} \\ \vdots & \\ s^{1/m} & \{X(s)\} \\ 1 & \{X(s)\} \end{Bmatrix}$$

$$\{\widetilde{F}(s)\} = \begin{Bmatrix} [0] \\ [0] \\ \vdots \\ [0] \\ (1 + bs^{q/m})\{F(s)\} \end{Bmatrix}$$

With $\{F(s)\} = 0$, the problem is now in terms of real, square, symmetric matrices. Thus, the eigenvalues will be distinct and either real or occur in complex conjugate pairs. Also, the eigenvectors will be orthogonal to one another. It is a straightforward matter to decouple the expanded equations of motion using standard techniques [1:67-68] Notice that for an n degree-of-freedom structure, the order of the expanded equations is $n(2m + q)$. From Equation 28, it can be seen that there are $(2m + q)$ branches to the problem, with n eigenvalues on each, resulting in $n(2m + q)$ eigenvalues. In a standard viscous formulation of the problem, only $2n$ eigenvalues would be found. The additional ones are due to the use of the fractional order derivatives. For a large structure, the higher order of the equations of motion represents a significant computational burden. Now that the existence of the solution has been proved, it will be beneficial to consider solution techniques that avoid solving the expanded equations of motion.

IV. Modified Matrix Iteration Solution

The current method of determining for the eigenstructure of the fractional order system deveoped in the previous chapter is to use a modified matrix iteration scheme on the homogeneous form of the original equation. Matrix iteration avoids computing and solving the characteristic polynomial of the matrix. Unlike using a Hessenberg matrix, which requires knowing the eigenvalue before the eigenvector can be calculated, matrix interation determines both at the same time.

Matrix iteration is typically used to find the eigenstructure of undamped systems. With some modification, the concept can be applied to damped systems. Two different algorithms will be needed to find all $n(2m+q)$ modes. For convenience, the modes on a given branch will be numbered beginning with the one corresponding to the eigenvalue with the smallest magnitude. A mode corresponding to an eigenvalue with larger magnitude will be referred to as a higher mode. Lower modes are defined in the same way.

For an undamped system, the homogeneous form of the equations of motion in the Fourier domain is

$$-\omega^2[M]\{\phi\}+[K]\{\phi\}=0 \tag{30}$$

or

$$[K]^{-1}[M]\{\phi\}=\frac{1}{\omega^2}\{\phi\} \tag{31}$$

To demonstrate matrix iteration, select a trial vector, $\{\psi\}$, and express it as a linear combination of the eigenvectors of $[K]^{-1}[M]$:

$$\{\psi\}=\sum_{i=1}^{n}c_i\{\phi_i\} \tag{32}$$

This is possible since the eigenvectors of $[K]^{-1}[M]$ span n-space. The only restriction on the c_i's is that $c_1 \neq 0$. Premultiplying both sides of Equation 32 by $[K]^{-1}[M]$ produces

$$[K]^{-1}[M]\{\psi\}=\sum_{i=1}^{n}\frac{c_i}{\omega_i^2}\{\phi_i\} \tag{33}$$

Subsequent multiplications produce

$$\left([K]^{-1}[M]\right)^k \{\psi\} = \sum_{i=1}^{n} \frac{c_i}{\omega_i^{2k}} \{\phi_i\} \tag{34}$$

Since for large k,

$$\omega_1^{2k} \ll \omega_2^{2k} \ll \cdots \ll \omega_n^{2k} \tag{35}$$

it is clear that Equation 34 converges to the lowest mode [10:124- 125]. If Equation 34 is normalized with respect to the same element between premultiplications by $[K]^{-1}[M]$, the the normalization factor reaches a constant value, equal to $1/\omega_1^2$ (since $c_1 \neq 0$), and the normalized vectors converge to the first mode. To find higher modes, subtract off lower modes using Turner's method [6:168-269]. Letting

$$[D] = [K]^{-1}[M] - \sum_{i=1}^{j-1} \frac{1}{\omega_i^2} \{\phi_i\}\{\phi_i\}^T[M] \tag{36}$$

then

$$[D]\{\phi\} = \frac{1}{\omega^2}\{\phi\} \tag{37}$$

converges to the j^{th} mode. Note that the lower modes must be normalized such that $\{\phi_i\}^T[M]\{\phi_i\} = 1$.

To apply this technique to a fractional order system, let $\lambda = s^{1/m}$. Then Equation 30 can be written as

$$\lambda^{2m}[M]\{\phi\} + [K(\lambda)]\{\phi\} = 0 \tag{38}$$

or

$$[K(\lambda)]^{-1}[M]\{\phi\} = \frac{-1}{\lambda^{2m}}\{\phi\} \tag{39}$$

where $[K(\lambda)]$ is equivalent to $[K(s)]$ in Equation 25. Each time the estimate of λ is updated, $[K(\lambda)]$ must be recomputed. Notice that for λ^{2m}, there are $2m$ possible values of λ. The different values arise because $z^{1/2m}$ is a multivalued function and has $2m$ branches. T' e value of λ on the k^{th} branch is computed using DeMoivre's Theorem [12:22]. Using the form $\lambda^{2m} = re^{i\theta}$,

$$\lambda = r^{1/2m}\left(\cos\frac{\theta + 2k\pi}{2m} + i\sin\frac{\theta + 2k\pi}{2m}\right) \tag{40}$$

The primary branch is assigned the number "0", so $k = 0, 1, 2, \ldots, 2m-1$.

Since the stiffness matrix is a function of λ, to find the higher modes Equation 36 must be modified:

$$[D(\lambda)] = [K(\lambda)]^{-1}[M] - \sum_{i=1}^{j-1} \frac{1}{\Lambda_i^{2m}} \{\Phi_i\}\{\Phi_i\}^T [M] \tag{41}$$

The quantities Λ_i and $\{\Phi_i\}$ are called pseudoeigenvalues and pseudoeigenvectors. They are computed from the eigenvector problem:

$$[K(\lambda)]^{-1}[M]\{\Phi\} = \frac{-1}{\Lambda^{2m}}\{\Phi\} \tag{42}$$

It is important to realize that the pseudoeigenvalues and pseudoeigenvectors are not modes of the system. Their computation is merely an intermediate step in calculating the solutions of the equations of motion. In computing the j^{th} mode of the system, only the first $j-1$ pseudomodes of Equation 42 are needed. Then Equation 41 is used to converge on the j^{th} mode of the system. Notice that for each new guess of λ, $j-1$ pseudoeigenvalues and pseudoeigenvectors must be recalculated. This repesents a significant computational burden. The next chapter proposes a technique to reduce the amount of computation required.

Note that this technique produces $2mn$ eigenvalues, but Equation 28 predicted $n(2m+q)$ eigenvalues. The remaining qn of the $n(2m+q)$ eigenvalues and eigenvectors are found using a scheme very similar to the one above [1:80-83]. After clearing the denominator of Equation 38, it can be written as

$$\begin{aligned} \lambda^{2m}(1+b\lambda^q)[M]\{\phi\} + (1+b\lambda^q)[K_E]\{\phi\} + \\ (E_0 + E_1\lambda^q)[K_V]\{\phi\} &= 0 \end{aligned} \tag{43}$$

Writing the equation in this form allows λ^q to appear explicitly in the equation, making it possible to find the remaining roots. Notice that these additional roots only exist for $b \neq 0$.

The solution method used to find the additional roots is somewhat subtle. By defining

$$Q = b\lambda^{2m+q} + \lambda^{2m} \tag{44}$$

$$[K'(\lambda)] = (1+b\lambda^q)[K_E] + (E_0 + E_1\lambda^q)[K_V] \tag{45}$$

Equation 43 can be written in the more recognizable form

$$[K'(\lambda)]^{-1}[M]\{\phi\} = \frac{-1}{Q}\{\phi\} \tag{46}$$

Matrix iteration is applied to this equation, with the i^{th} estimate of λ determined from

$$\lambda_i = \left[\left(\frac{Q - \lambda_{i-1}^{2m}}{b\lambda_{i-1}^{2m+q-1}}\right)^q\right]^{1/q} \tag{47}$$

The k^{th} branch of the q^{th} root of the quantity in brackets is used to determine the eigenvalue on that branch.

Turner's method is again employed to find the higher modes on each branch, as in Equation 41, with $Q(\lambda_i)$ replacing ω_i^2. A program which uses the above methods to compute the additional modes is given in Appendix D.

This chapter has shown that it is possible to find all $n(2m+q)$ eigenvalues and eigenvectors without solving the expanded equations of motion. However, the technique still requires a substantial amount of computation. In the next chapter, a technique is proposed which greatly reduces the computational burden.

V. Spectrum Shift Technique

While the modified matrix iteration technique is effective, it is not very efficient. In this chapter, spectrum shift methods will be combined with the matrix iteration technique reducing the amount of computation required. The purpose of spectrum shift is to shift the eigenvalues of the system so that the desired eigenvalue becomes the fundamental one. Matrix iteration will then produce the desired eigenvalue. If spectrum shift methods could be used to compute the higher modes in the viscoelastic model, the pseudoeigenvalues and pseudoeigenvectors of the corresponding $[K(\lambda)]^{-1}[M]$ would not have to be computed. Determining the appropriate spectrum shifts is not easy, and requires certain precautions, which will be presented later.

The spectrum shift technique is usually used in elastic systems when a particular frequency and corresponding mode shape are of interest. To illustrate the theory behind the spectrum shift technique, consider again an undamped system

$$[[K] - \omega^2[M]]\,\{\phi\} = 0 \tag{48}$$

Picking the shift factor, μ, close to the desired ω_i^2 gives the shifted equations [8:330]

$$\left[[K] - \mu[M] - (\omega^2 - \mu)[M]\right]\{\phi\} = 0 \tag{49}$$

Letting

$$[\widehat{K}] = [K] - \mu[M] \quad \text{and} \quad \widehat{\omega^2} = \omega^2 - \mu \tag{50}$$

Then

$$\left[[\widehat{K}] - \widehat{\omega^2}[M]\right]\{\phi\} = 0 \quad \text{or} \quad [\widehat{K}]^{-1}[M]\{\phi\} = \frac{1}{\widehat{\omega^2}}\{\phi\} \tag{51}$$

Applying matrix iteration to this equation produces the mode closest to μ.

Now consider the matrix $[K(\lambda_i)]^{-1}[M]$ of the viscoelastic model. Only the i^{th} eigenvalue and eigenvector are desired. By letting $\lambda = s^{1/m} = (i\omega)^{1/m}$, Equations 50 and 51 can be written as

$$\begin{aligned} \widehat{\lambda^{2m}} &= \lambda_i^{2m} + \mu \\ [\widehat{K}(\lambda_i)] &= [K(\lambda_i)] - \mu[M] \\ [\widehat{K}(\lambda_i)]^{-1}[M]\{\phi\} &= \frac{-1}{\widehat{\lambda^{2m}}}\{\phi\} \end{aligned} \tag{52}$$

As a first guess of the appropriate shift factor for the i^{th} mode, the eigenvalue of $[K(\lambda_{i-1})]^{-1}[M]$ closest to λ_{i-1} is used. It is computed by using Turner's method. The dynamical matrix is

$$\begin{aligned} [D(\lambda_{i-1})] &= [\widehat{K}(\lambda_{i-1})]^{-1}[M] - \frac{1}{\lambda_{i-1}^{2m}}\{\phi_{i-1}\}\{\phi_{i-1}\}^T[M] \\ [D(\lambda_{i-1})]\{\phi\} &= \frac{1}{\widehat{\lambda^{2m}}}\{\phi\} \end{aligned} \tag{53}$$

If μ_{i-1} was the shift used to find λ_{i-1}, then by Equation 52, the new shift factor is

$$\mu_i = \mu_{i-1} - \widehat{\lambda^{2m}} \tag{54}$$

Since the magnitude of the i^{th} eigenvalue must be larger than the magnitude of λ_{i-1}, if

$$|\mu_i| < |\mu_{i-1}| \tag{55}$$

then $\widehat{\lambda^{2m}}$ was in the wrong direction. The shift is recomputed as

$$\mu_i = \mu_{i-1} + \widehat{\lambda^{2m}} \tag{56}$$

Notice that matrix iteration on

$$[\widehat{K}(\widehat{\lambda}_i, \mu_i)]^{-1}[M] = \frac{1}{\widehat{\lambda_i^{2m}}}\{\phi_i\} \tag{57}$$

will converge to the $i-1$ mode if the magnitude of μ_i is not large enough. If this occurs, μ_i is adjusted by adding the new $\widehat{\lambda^{2m}}$ (as in Equation 56). A program employing these techniques is listed in Appendix C.

For undamped systems, the j^{th} eigenvalues on all $2m$ branches have the same magnitude and are evenly spaced on a circle about the origin. For lightly damped systems, the j^{th} eigenvalues lie

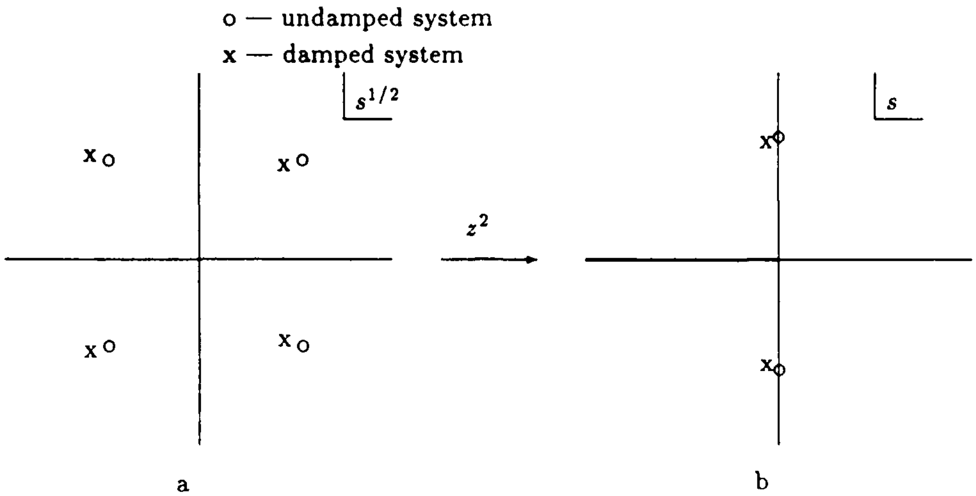

Figure 2. Locations of λ and λ^2 values

near the those for an undamped system. This is portrayed graphically for a single degree of freedom system with $\alpha = 1/2$ in Figure 2a. Since λ^{2m} is roughly the same magnitude for all the eigenvalues, the program in Appendix C can be modified slightly to use the λ_i^{2m} values on the principal branch to calculate shifts for the other branches. This modification is valid for systems with less than a 0.01 damping ratio.

To understand the location of the λ^2 values in the s-plane, it is necessary to realize that the Riemann surface for the function $w = z^{1/2}$ consists of two Riemann sheets, joined together at the branch cut. Taking the branch cut along the negative real axis, the sheets can be defined by

$$\begin{aligned} S_0 &= \{z | -\pi \leq \arg(z) < \pi\} \\ S_1 &= \{z | \pi \leq \arg(z) < 3\pi\} \end{aligned} \tag{58}$$

So the eigenvalues in the first and fourth quadrants of the $s^{1/2}$-plane map into the second and third quadrants, respectively, of S_0. These are shown in Fiqure 2b. But the eigenvalues in the second and third quadrants of the $s^{1/2}$-plane map into the fourth and first quadrants, respectively, of S_1. To

see this let $re^{i(3\pi/4+\delta)}$ represent the second quadrant eigenvalue, where δ is an small angle. Then

$$\arg(\lambda^2) = \frac{3\pi}{2} + 2\delta \tag{59}$$

Since this angle is greater than π, λ^2 is on S_1 at the angle given by Equation 59.

The third quadrant eigenvalue is a little more subtle. Its angle is $-(3\pi/4 + \delta)$, so

$$\arg(\lambda^2) = -\frac{3\pi}{2} - 2\delta \tag{60}$$

But neither sheet contains values with this angle. When the value crossed the negative real axis in the negative direction, its angle experienced a 4π jump discontinuity from $-\pi$ to 3π. Therefore the angle is really

$$\arg(\lambda^2) = -\frac{3\pi}{2} - 2\delta + 4\pi = \frac{5\pi}{2} - 2\delta \tag{61}$$

This angle is in the first quadrant of S_1. Notice that for undamped systems, the λ^2 values in S_1 lie directly above those in S_0. To map back into the $s^{1/2}$-plane, the 4π must be subtracted off before taking the square root.

For a ten degree-of-freedom system, the spectrum shift technique more than halved the computation time required by the modified matrix iteration technique. Storing the principal branch's λ^{2m} values reduced the computation time by another 50%. (Exact computation times are given in the next chapter.) Computed eigenvalues were accuratc to at least five significant figures.

VI. *Example Problem*

To demonstrate the efficiency of this technique, a ten degree-of-freedom model was considered. The rod was similar to the one in Figure 1, and its equations of motion had the same form. The rod was assumed to be pure aluminum, with Butyl B252 damping pads. The values of the parameters were [4](all values are in compatible mks SI units)

$$\begin{aligned} \rho &= 2.71 \cdot 10^3 & E &= 5.516 \cdot 10^{10} \\ A &= 0.0625 & G_0 &= 7.6 \cdot 10^5 \\ A_i &= 0.0625 & G_1 &= 2.95 \cdot 10^5 \\ L &= 0.909 & b &= 0.001 \\ t_i &= 0.1 \end{aligned} \tag{62}$$

These parameters resulted in low damping, on the order of 10^{-2}, so it could be solved using the modified spectrum shift technique, as well as by using modified matrix iteration or spectrum shift. The computation times for two different pad thicknesses are given in Table 1. The solution took longer than for the thinner pad due to the increased damping.

The damping in the system was increased by decreasing the thickness of the viscoelastic pads to 0.01m. For this case, the equivalent damping ratio was 0.069, as computed from the fundamental mode. The eigenvalues and eigenvectors for the spectrum shift solution are listed in Appendix E. For completeness, the additional roots (computed using modified matrix iteration) are also included in Appendix E. For the principal branch, the complex frequencies and mode shapes were found to be

Technique	$t = 0.1$m	$t = 0.05$m
Modified matrix iteration	0:52.11	1:14.51
Spectrum shift	0:21.78	0:32.43
Modified spectrum shift	0:12.06	0:15.15

Table 1. Computation Times (in CPU minutes)

$$
\left\{
\begin{array}{rcr}
-107 & + & 1545i \\
-77 & + & 2962i \\
-66 & + & 4459i \\
-61 & + & 6051i \\
-60 & + & 7762i \\
-61 & + & 9606i \\
-64 & + & 11566i \\
-69 & + & 13567i \\
-74 & + & 15415i \\
-79 & + & 16779i
\end{array}
\right\}
\tag{63}
$$

and

$$
\begin{bmatrix}
1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\
1.9 & 1.7 & 1.3 & 0.8 & 0.3 & -0.3 & -0.8 & -1.3 & -1.7 & -1.9 \\
2.7 & 1.8 & 0.7 & -0.3 & -0.9 & -0.9 & -0.3 & 0.7 & 1.8 & 2.7 \\
3.2 & 1.4 & -0.4 & -1.1 & -0.5 & 0.6 & 1.1 & 0.4 & -1.4 & -3.2 \\
3.5 & 0.5 & -1.2 & -0.6 & 0.8 & 0.8 & -0.6 & -1.2 & 0.5 & 3.5 \\
3.5 & -0.5 & -1.2 & 0.6 & 0.8 & -0.8 & -0.6 & 1.2 & 0.5 & -3.5 \\
3.2 & -1.4 & -0.4 & 1.1 & -0.5 & -0.6 & 1.1 & -0.4 & -1.4 & 3.2 \\
2.7 & -1.8 & 0.7 & 0.3 & -0.9 & 0.9 & -0.3 & -0.7 & 1.8 & -2.7 \\
1.9 & -1.7 & 1.3 & -0.8 & 0.3 & 0.3 & -0.8 & 1.3 & -1.7 & 1.9 \\
1.0 & -1.0 & 1.0 & -1.0 & 1.0 & -1.0 & 1.0 & -1.0 & 1.0 & -1.0
\end{bmatrix}
\tag{64}
$$

The first three mode shapes are plotted in Figures 3 to 5. The magnitude of the complex frequencies for the first five modes is less than 10% higher than those for an undamped continuum model (refer to Appendix B for a description of the continuum model), but the higher frequencies differ by up to 20%.

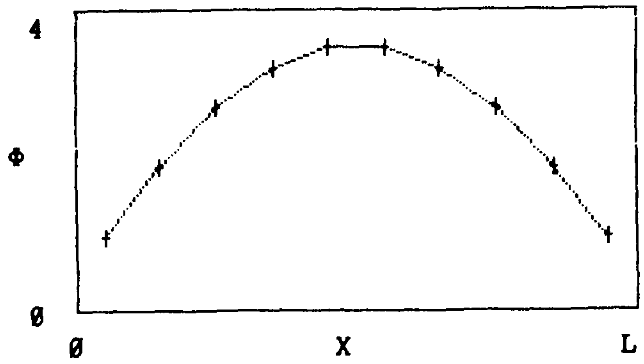

Figure 3. First Mode Shape for Damped Rod

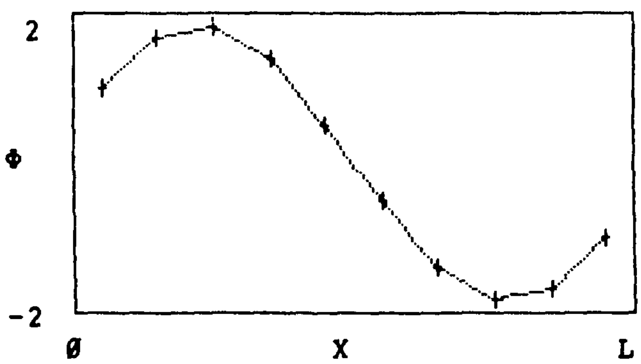

Figure 4. Second Mode Shape for Damped Rod

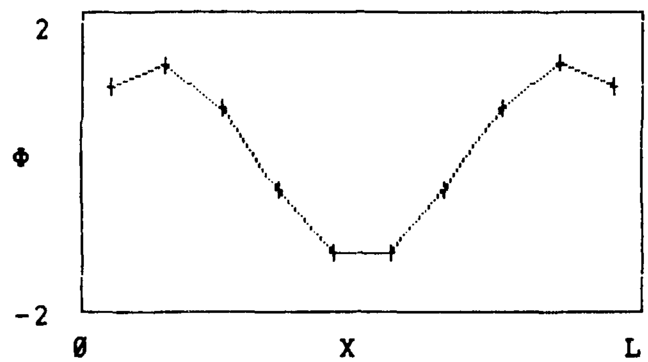

Figure 5. Third Mode Shape for Damped Rod

The spectrum shift method complements the finite element model. With spectrum shift, finite element problems with viscoelastic damping can be solved much faster than with modified matrix iteration. For a ten degree-of- freedom model, the savings was more than 50% of the CPU time.

VII. Conclusions and Recommendations

The spectrum shift technique is more efficient than the matrix iteration technique. The computational burden does not increase as drastically with increasing degrees of freedom. For lightly damped systems, the modified spectrum shift technique represents even greater computational savings.

The existing program can be made more efficient by realizing that in real systems eigenvalues and eigenvectors appear in complex pairs, and by taking advantage of the symmetry of the stiffness and mass matrices. Also, for larger systems, it would be beneficial to examine matrix inversion techniques that are designed to handle large matrices.

Spectrum shift techniques were attempted on the solution for the additional qn eigenvalues due to $b \neq 0$, but the initial results were discouraging. The eigenvalues are all close in magnitude, which presents a challenge to the spectrum shift method. Time constraints prevented a closer look into this approach, but the results presented in the last chapter suggest it would be worth while to look into this some more, especially for large m.

Bibliography

1. Bagley, R. L., *Applications of Derivatives to Viscoelasticity*, Ph. D. Dissertation, Air Force Institute of Technology; also published as Air Force Materials Laboratory TR-79-4103, Nov. 1979 (AD- A071726).

2. Bagley, R. L. and Torvik, P. J., "Fractional Calculus—A Different Approach to the Analysis of Viscoelastically Damped Structures," *AIAA Journal*, Vol. 21, May 1983, pp. 741-748.

3. Bagley, R. L. and Torvik, P. J., "On the Fractional Calculus Model of Viscoelastic Behavior," *Journal of Rheology*, Vol. 30, No. 1, 1986, pp. 133-155.

4. Bagley, R. L. and Torvik, P. J., "Fractional Calculus in the Transient Analysis of Viscoelastically Damped Structures," *AIAA Journal*, Vol. 23, June 1985, pp. 918-925.

5. Bagley, R. L. and Torvik, P. J., "A Theoretical Basis for the Application of Fractional Calculus to Viscoelasticity," *Journal of Rheology*, Vol. 27, No. 3, 1982, pp. 201-210.

6. Bisplinghoff, R. L., Ashley, H., and Halfman, R. L., *Aeroelasticity*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1955.

7. Christensen, R. M., *Theory of Viscoelasticity: An Introduction*, Academic Press, New York, 1971, p. 42.

8. Craig, R. R., Jr., *Structural Dynamics: An Introduction to Computer Methods*, John Wiley & Sons, New York, 1981.

9. Gel'fand, I. M., and Shilov, G. E. *Generalized Functions, Vol I*, Academic Press, New York, 1964.

10. Hurty, W. C. and Rubinstein, M. F., *Dynamics of Structures*, Prentice-Hall, Inc. , Englewood Cliffs, New Jersey, 1964.

11. Jenkins, W. M., *Matrix and Digital Computer Methods in Structural Analysis*, McGraw-Hill, London, 1969, pp. 170-171.

12. Spiegel, M. R., *Mathematical Handbook of Formulas and Tables*, McGraw-Hill Book Company, New York, 1968.

13. Torvik, P. J. and Bagley, R. L., "On the Appearance of the Fractional Derivative in the Behavior of Real Materials," *Journal of Applied Mechanics*, Vol. 51, No. 2, 1984.

14. Torvik, P. J. and Bagley, R. L., "Fractional Derivatives in the Description of Damping Materials and Phenomena," *The Role of Damping in Vibration and Noise Control*, DE-Vol. 5, The American Society of Mechanical Engineers.

Appendix A. *Programming Flowcharts and Special Techniques*

This appendix presents flowcharts for the three programs included in this thesis in Figures 6 through 8. The first one is for the program in Appendix D, VDRMI, which calculates the first $2mn$ modes of a viscoelastically damped rod using modified matrix iteration. The second flowchart is for the program in Appendix E, NONZEROB, which calculates the additional modes due to $b \neq 0$ by modified matrix iteration. Notice that its flow chart is very similar to the one for VDRMI. The third flow chart is for the program in Appendix C, VDRSS1, which computes the first $2mn$ modes by the spectrum shift technique presented in Chapter V. The programs are similar in the logic used in each one. Both VDRMI and VDRSS1 calculate the new guess of λ from λ^2 by Equation 40, while NONZEROB uses Equation 47. If λ is within tolerance, then λ and $\{\phi\}$ are printed out. If λ is not within tolerance, it is used to compute $[D(\lambda)]$ (See Equation 41 and following text for VDRMI and NONZEROB, Equation 52 for VDRSS1).

In VDRMI and NONZEROB, if this is not the fundamental mode on the current branch, then the pseudomodes (discussed in Chapter IV) must be computed and subtracted off by Equation 41.

The inverse of $[K(\lambda)]$ is needed to compute $[D(\lambda)]$. This inversion is carried out by first expressing $[K(\lambda)]$ in terms of a Choleski decomposition [11:170]:

$$[K(\lambda)] = [U]^T[U]$$

This is valid as long as $[K(\lambda)]$ is symmetric. The inverse of $[K(\lambda)]$ is given by

$$[K(\lambda)]^{-1} = [U]^{-1}([U]^{-1})^T$$

Once $[D(\lambda)]$ is determined, a new estimate of λ and $\{\phi\}$ can be computed. The process is repeated until λ is within the desired tolerance.

The new estimate of λ and $\{\phi\}$, and the pseudomodes in VDRMI and NONZEROB, are determined using matrix iteration. Convergence in the matrix iteration portions of the programs was

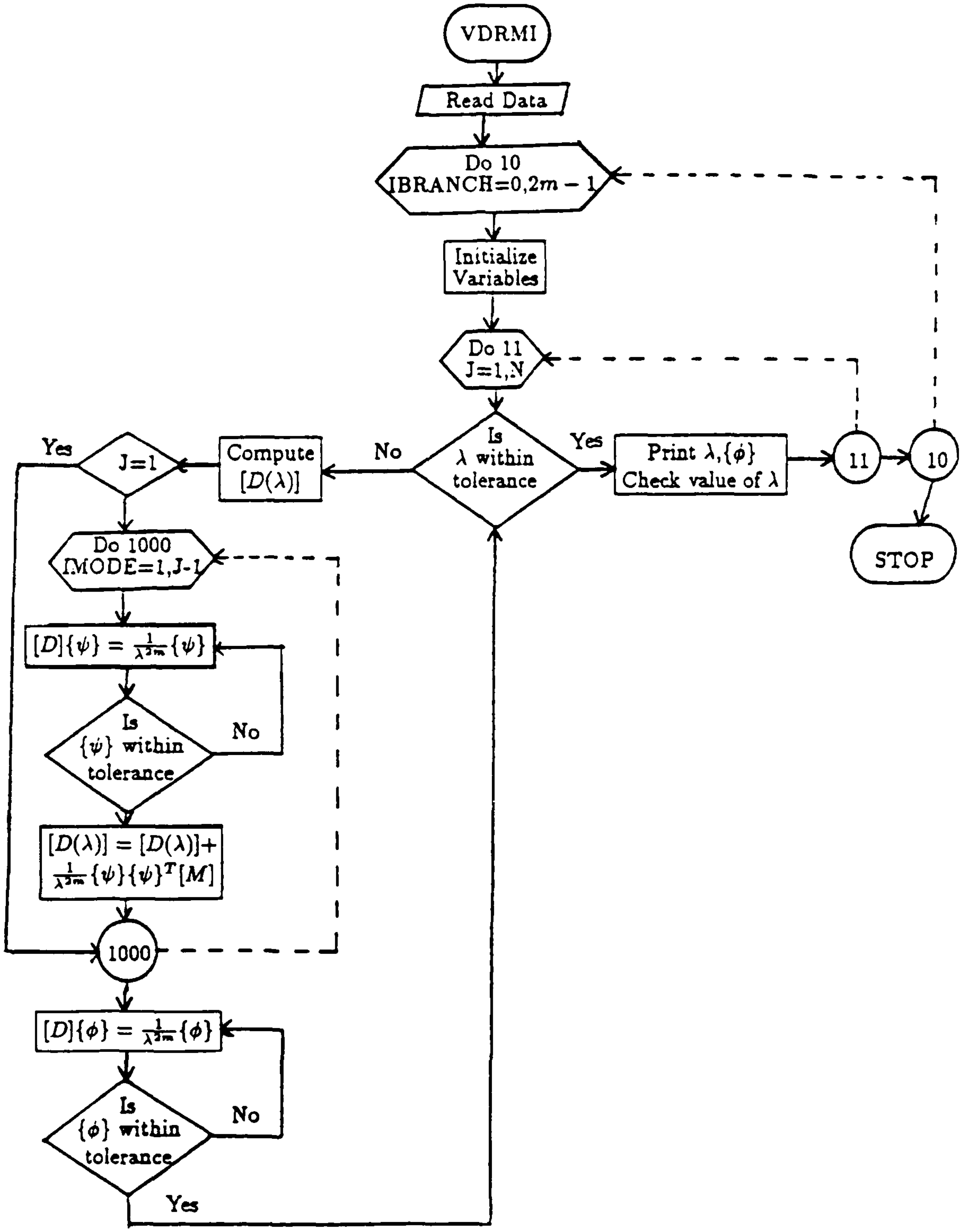

Figure 6. Flowchart for VDRMI

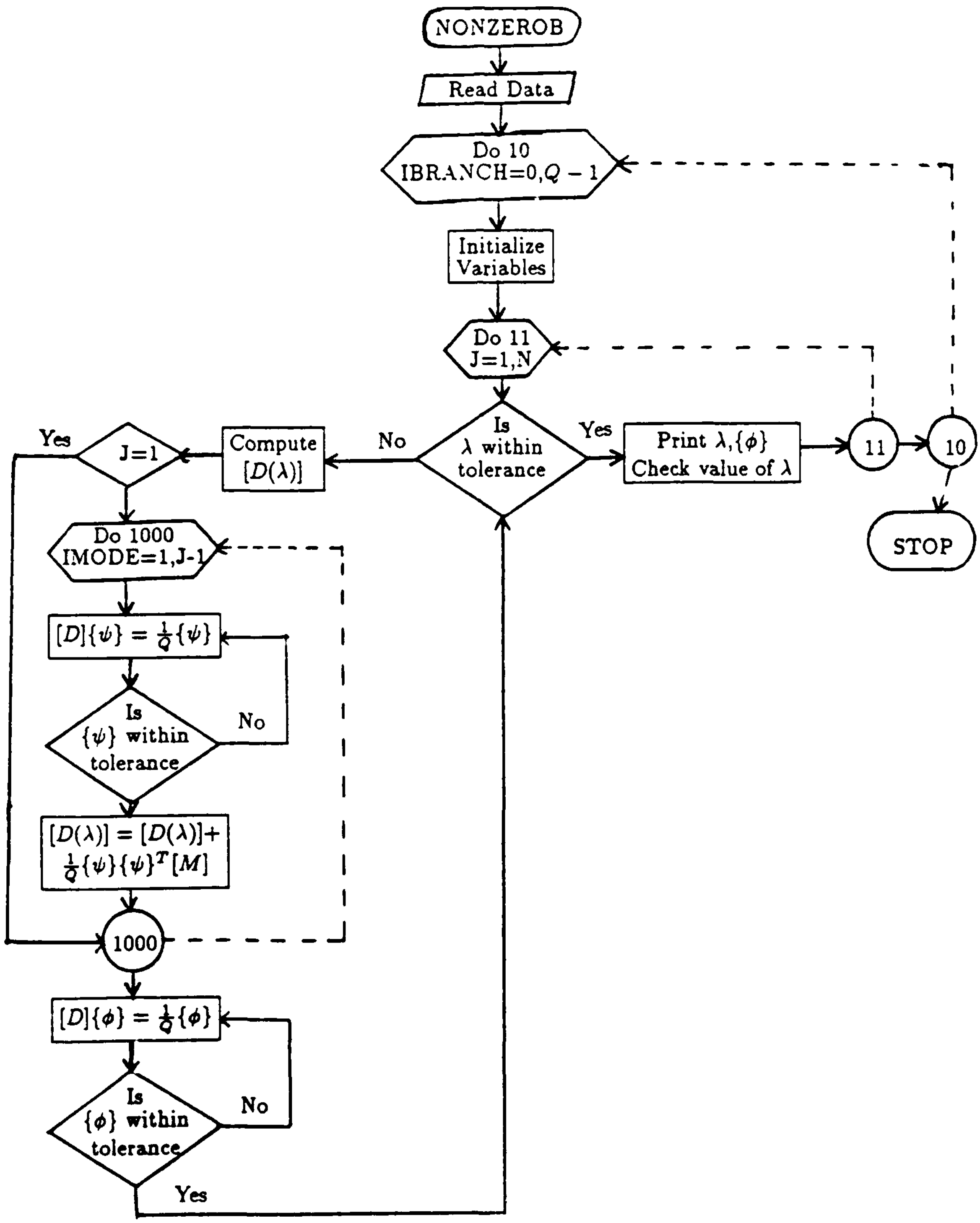

Figure 7. Flowchart for NONZEROB

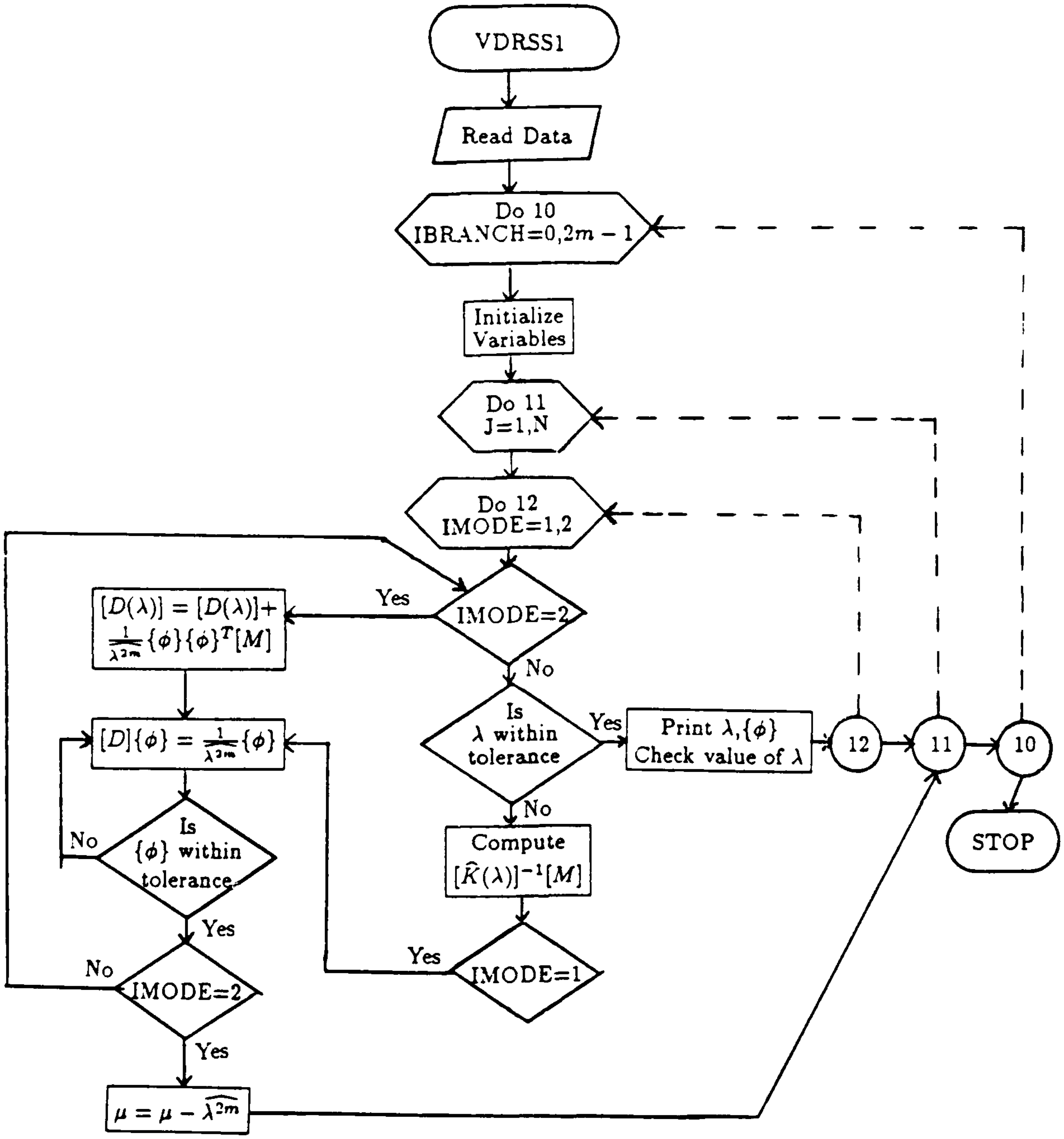

Figure 8. Flowchart for VDRSS1

determined by comparing the norm of the difference of successive guesses. Convergence occurred when this norm was below a desired tolerance. Clearly, as n increases, the tolerance on $\{\phi\}$ does not have to be as tight for the same accuracy in λ. In VDRMI and NONZEROB, the tolerance values were picked through trial and error to find ones that gave convergence with the desired accuracy in λ. In VDRSS1, the tolerance for $\{\phi\}$, ϵ_1, was related to the tolerance of λ, ϵ_0, by

$$\epsilon_1 = \epsilon_0^{1/m}\sqrt{n}$$

After the j^{th} eigenvalue and eigenvector are found in VDRSS1, the shift factor for the $j+1$ eigenvalue must be computed. This is done by applying Equations 53 through 56.

Once an eigenvalue and eigenvector have been computed, they are put back into the expanded equations of motion to check the accuracy of the solution. To make the equaitons of motion more manageable, multiply Equation 29 by $\{\widetilde{\phi}\}$. Setting $\{F(s)\} = 0$, $s^{1/m} = \lambda$, and replacing $\{\phi\}$ by $\{X(s)\}$,

$$\lambda_c = -\frac{\{\widetilde{\phi}\}^T[\widetilde{K}]\{\widetilde{\phi}\}}{\{\widetilde{\phi}\}^T[\widetilde{M}]\{\widetilde{\phi}\}} \tag{65}$$

Expanding,

$$\lambda_c = \frac{[(2m+q-1)b\lambda^{2m+q} + (2m-1)\lambda^{2m}]\{\phi\}^T[M]\{\phi\} + (q-1)\lambda^q\{\phi\}^T[A_q]\{\phi\} - \{\phi\}^T[A_q]\{\phi\}}{[(2m+q)b\lambda^{2m+q-1} + 2m\lambda^{2m-1}]\{\phi\}^T[M]\{\phi\} + q\lambda^{q-1}\{\phi\}^T[A_q]\{\phi\}} \tag{66}$$

This is a convenient check on the computed value of λ, and its value is printed out right after λ so that a direct comparison can be made. With the proper ϵ values, accuracy to 5 significant digits was typical for all programs.

Appendix B. *Correlation of FORTRAN Program for Purely Elastic Rod*

This appendix contrasts the first three mode shapes of a ten degree-of- freedom finite element model against those of a continuum model for a purely elastic rod. This will give an indication of the accuracy of the model for the viscoelastically damped case.

The continuum model for a purely damped elastic rod is developed from the harmonic equation

$$\rho A \frac{\partial^2 u}{\partial t^2} = EA \frac{\partial^2 u}{\partial x^2} \tag{67}$$

with boundary conditions

$$u(0,t) = u(L,t) = 0$$

The resonant frequencies of the system are found to be

$$\omega = \frac{n\pi}{L}\sqrt{\frac{E}{\rho}} \qquad n = 1, 2, \ldots \tag{68}$$

The mode shapes are

$$\phi(x) = \sin(\frac{n\pi}{L}x) \tag{69}$$

In Figures 9 through 11, the first few modes of the continuum model are plotted against the computed modes of a ten degree-of- freedom finite element model. The modes were computed using the program in Appendix C, with the viscoelastic term set to zero. These comparisons demonstrate the validity of the program for elastic systems.

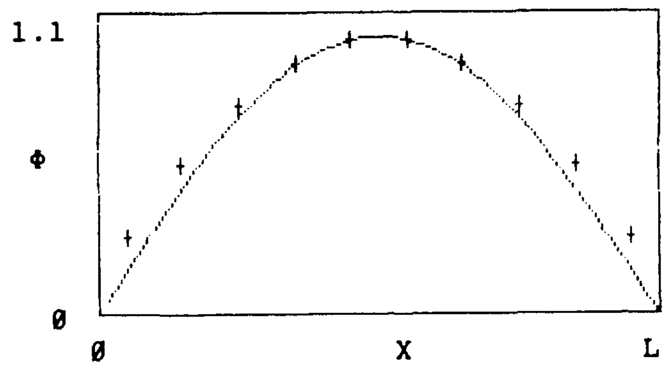

Figure 9. First Mode Shape for Undamped Rod

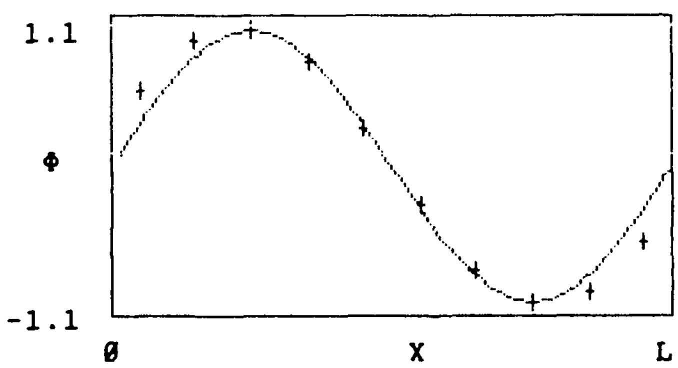

Figure 10. Second Mode Shape for Undamped Rod

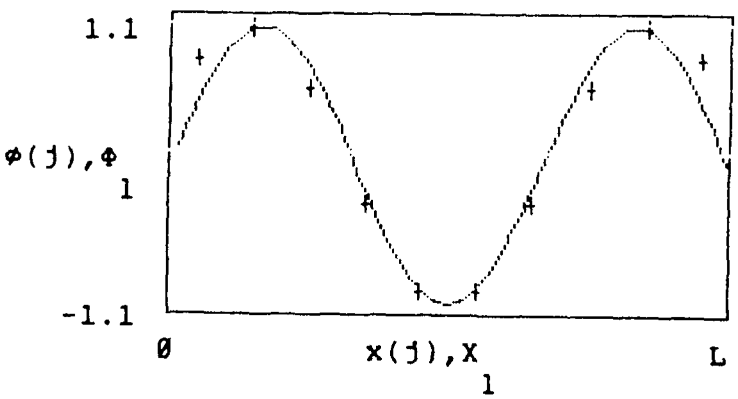

Figure 11. Third Mode Shape for Undamped Rod

Appendix C. *FORTRAN Program for Spectrum Shift Technique*

This computer program calculates the first $2mn$ eigenvectors and eigenvalues for a viscoelastically damped rod using the spectrum shift technique presented in Chapter V. A flowchart for this program is presented in Appendix A. The program uses unformatted READ and WRITE statements, which may produce output different from that shown in Appendix F on computers other than a VAX/VMS. Some of the input parameters are the number of degrees of freedom and the physical characteristics of the rod and pads, including the mass and stiffness matrices.

```
      PROGRAM VDRSS1
C
C     FORTRAN Code for viscoelastically damped rod using matrix
C     iteration with spectrum shift.
C
C     Set up variables needed in program
C     Declaration statements for variables.
C
      INTEGER IQ,IM,IMAX1,IMAX2,IMAX3,N
      REAL A,ALPHA,B,E0,E1,EPS1,LE,MAG,RHOE
      COMPLEX IMRIO,IMRIOC,IOMA,IOMAL,IOMAS,ITA,IOM2H,IOM2S,IOMEGA,
     1        PTAOP,PTAQP,PTMP,PROD,SUM,
     2                                 IOM2,MU,MUS,
     3        KE(10,10),KV(10,10),M(10,10),
     4        D(10,10),ERR(10),K0(10,10),K1(10,10),
     4        KINV(10,10),KOMEGA(10,10),PHIG(10),PHILG(10),PPTM(10,10),
     4        PSI(10),PSIG(10),SHIFT(10),U(10,10),UINV(10,10),
     5                                             PHI(10)
C
C     N = Number of Degrees of Freedom
C     IMAX = Maximum number of times to run iteration loop
C     KV = Stiffness matrix for viscoelastic damping material
C     KE = Stiffness matrix for elastic rod
C     M = Mass Matrix of total structure
C     LE = Length of one element
C     A = Cross-sectional area of rod
C     RHO = material density
C     E = Young's modulus for elastic material.
C     E0,E1,B,ALPHA = Parameters of Young's modulus for viscoelastic
C                     material.
C     IOM2 = (I*OMEGA)**2 = eigenvalue
C     IOMEGA = i * system frequency (sometimes also referred to
C              as an "eigenvalue"
C     PHI = eigenvector
C     PHIG = guess at an eigenvector
C     PHILG = last guess at an eigenvector
C     D = Dynamical matrix
C     EPS = Tolerance level

C     Open input and output data files.

      OPEN (UNIT=5,FILE='INPUT',STATUS='OLD')
      OPEN (UNIT=9,FILE='DEBUG',STATUS='NEW')
      OPEN (UNIT=10,FILE='OUTPUT1',STATUS='NEW')

      write (10,*) 'OUTPUT FOR VDRSS1'

C     Set value of PI
      PI = 3.141592654

C     Read in EPS, RHO, A, LE, parameters for E
```

```
C
      READ (5,*) EPS0
      READ (5,*) RHOE
      READ (5,*) AE
      READ (5,*) LE
      READ (5,*) E
      READ (5,*) E0,E1,B

      write (10,*) 'rhoe',rhoe
      write (10,*) 'ae',ae
      write (10,*) 'le',le
      write (10,*) 'e',e
      write (10,*) 'e0',e0,'  e1',e1,'  b',b

      READ (5,*) N, IQ, IM
      ALPHA  = REAL(IQ)/REAL(IM)
      READ (5,*) IMAX1,IMAX2,IMAX3

      write (10,*) 'n',n,' iq',iq,' im',im,' alpha',alpha
      write (10,*) 'imax1',imax1,'  imax2',imax2,'  imax3',imax3

      EPS1 = (EPS0**(1/REAL(IM)))*SQRT(REAL(N))

      write (10,*) 'eps0',eps0,'  eps1',eps1

C     Read in stiffness and mass matrices for both elastic and
C     viscoelastic materials.

      DO 40 ICOL = 1,N
          DO 50 IROW = 1,N
              READ (5,*) KE(IROW,ICOL)
      write (10,*)'ke',irow,icol,ke(irow,icol)

            KE(IROW,ICOL) = KE(IROW,ICOL)*E*AE/LE
      write (10,*)'ke',irow,icol,ke(irow,icol)

50        CONTINUE
40    CONTINUE

      DO 42 ICOL = 1,N
          DO 52 IROW = 1,N
              READ (5,*) KV(IROW,ICOL)

      write (10,*)'kv',irow,icol,kv(irow,icol)

C     Compute K0 and K1 elements.

              K0(IROW,ICOL)   = E0*KV(IROW,ICOL)
              K1(IROW,ICOL)   = E1*KV(IROW,ICOL)

52        CONTINUE
```

```
42    CONTINUE

      DO 43 ICOL = 1,N
           DO 53 IROW = 1,N
                READ (5,*) M(IROW,ICOL)
      write (10,*)'m',irow,icol,m(irow,icol)

              M(IROW,ICOL) = M(IROW,ICOL)*RHOE*AE*LE/6.
      write (10,*)'m',irow,icol,m(irow,icol)
53         CONTINUE
43    CONTINUE

C****************************************************************
C*          Calculate roots for branch = IBRANCH               *
C****************************************************************

      DO 10 IBRANCH = 0,2*IM-1

C          Set initial values of iom2h and mu

           IOM2H = CMPLX(0.1,0.1)
         IOM2S = 0.0
         IOMAS = 0.0
         MU = 0.0
         MUS = 0.0
           DO 11 J = 1,N

         DO 12 IMODE = 1,2

           write (9,*) 'Computing Eigenvalue ',j,' on branch ',ibranch,
     1                 ' imode = ',imode
           print *, 'Computing Eigenvalue ',j,' on branch ',ibranch,
     1                 ' imode = ',imode

C          Set initial eigenvector guess.

           DO 15 I = 1,N
                PHIG(I) = 1.0/I
15         CONTINUE

      DO 20 IGUESS1 = 1,IMAX1

C     Check if this is the second iteration for current value of mu --
C     if so, use current value of ioma and skip right to computation of
C     new mu.

      IF(IMODE .EQ. 2) GOTO 3000


C     Calculate IOMA = (i*omega)**alpha = ((i*omega)**2)**alpha/2
C                = (iom2h-mu)**alpha/2
```

```
C              = (abs(iom2h-mu)*exp(i*ang))**alpha/2
C              = ((mag**(1/im))*exp(i*ang*(1/2*im))**iq = IMRIO ** iq

          IOM2 = IOM2H - MU

C     Check each new value of IOM2 -- its magnitude should be greater
C     than the previous value of MU.

        IF (ABS(IOM2) .LT. ABS(MUS)) THEN
           MUS = MU
           MU = MU + IOM2H

      write (9,*) 'abs(iom2)',abs(iom2),'abs(mus)',abs(mus)
      print *, 'mus 1st chk',mus,'iom2h',iom2h

           IOM2H = 0.0
C         Reset initial eigenvector guess.

          DO 17 I = 1,N
             PHIG(I) = 1.0/I
17        CONTINUE
           GOTO 20
        ENDIF

          REIOM2  = REAL(IOM2)
          AIMIOM2  = AIMAG(IOM2)
          MAG     = SQRT(REIOM2**2 + AIMIOM2**2)
          ANG     = ATAN2(AIMIOM2,REIOM2)

      IF (ANG .LT. 0) ANG = ANG + 2*PI

          ARG = (ANG + 2*IBRANCH*PI)/REAL(2*IM)

          IMRIO = (MAG**(1/REAL(2*IM)))*CMPLX(COS(ARG),SIN(ARG))

        IOMEGA = IMRIO**IM

          IOMA = IMRIO**IQ

        DIF = ABS((IOMA - IOMAS)/IOMA)

        IF (DIF .LT. 2.*SQRT(EPS0)) THEN
             IF(IOM2H .EQ. 0) GO TO 29
           MUS = MU
           MU = MU + IOM2H

      print *, 'mus',mus,'iom2h',iom2h
      write (9,*) 'mus',mus,'iom2h',iom2h

           IOM2H =0.0
C         Reset initial eigenvector guess.
```

```
                DO 16 I = 1,N
                        PHIG(I) = 1.0/I
16              CONTINUE

              GOTO 20
          ENDIF

            DIF = ABS((IOMA - IOMAL)/IOMA)

            write (9,*) 'dif',dif,'errnorm',errnorm

            IF (DIF .LT. EPS0 .AND. ERRNORM .LT. EPS1) GO TO 5400
            IOMAL = IOMA

C     Now compute K(omega).

29    DO 30 ICOL = 1,N
           DO 31 IROW = 1,N
             KOMEGA(IROW,ICOL) = KE(IROW,ICOL) +
     1       (K0(IROW,ICOL) + IOMA*K1(IROW,ICOL))/(1 + B*IOMA)
     2       - MU*M(IROW,ICOL)
31         CONTINUE
30    CONTINUE


C     ********************************************************
C     *                                                      *
C     *              Compute dynamical matrix                *
C     *                                                      *
C     ********************************************************
C
C          First compute inverse of K by using a Cholesky decomposition
C               KOMEGA = Utranspose * U
C          Compute U
           U(1,1) = CSQRT(KOMEGA(1,1))

           DO 110 ICOL = 2,N
                U(1,ICOL) = KOMEGA(1,ICOL)/U(1,1)
110        CONTINUE

           DO 120 IROW = 2,N
                SUM = 0.0
                DO 130 ITER = 1,IROW-1
                     SUM = SUM + U(ITER,IROW)**2
130             CONTINUE
                U(IROW,IROW) = CSQRT(KOMEGA(IROW,IROW)-SUM)

                DO 140 ICOL = IROW+1,N
                    SUM = 0.0
                    DO 150 ITER =1,IROW-1
```

```
                          SUM = SUM + U(ITER,IROW)*U(ITER,ICOL)
150                   CONTINUE
                      U(IROW,ICOL) = (KOMEGA(IROW,ICOL)-SUM)/U(IROW,IROW)

140               CONTINUE
120           CONTINUE
C             inverse of KOMEGA = (inverse of U)*(inverse of U,transposed)
C                 First calculate inverse of U
                  DO 200 ICOL = 1,N
                      UINV(ICOL,ICOL) = 1.0/U(ICOL,ICOL)

                      DO 210 IROW = 1,ICOL-1
                          SUM = 0.0
                          DO 220 ITER = IROW, ICOL-1
                              SUM = SUM + UINV(IROW,ITER)*U(ITER,ICOL)
220                       CONTINUE
                          UINV(IROW,ICOL) = -SUM/U(ICOL,ICOL)

210                   CONTINUE
200               CONTINUE
C                 Now for KOMEGA inverse
                  DO 230 IROW = 1,N
                      DO 240 ICOL =1,N
                      KINV(IROW,ICOL) = 0.0
                          DO 250 ITER = 1,N
                             KINV(IROW,ICOL) = KINV(IROW,ICOL) +
    1                         UINV(IROW,ITER) * UINV(ICOL,ITER)
250                       CONTINUE
240                   CONTINUE
230               CONTINUE
C             Compute D = KINV * M
              DO 500 IROW = 1,N
                  DO 600 ICOL = 1,N
                    D(IROW,ICOL) = 0.0
                    DO 700 ITER = 1,N
                      D(IROW,ICOL) = D(IROW,ICOL) +
    1                                KINV(IROW,ITER) * M(ITER,ICOL)
700                 CONTINUE
600               CONTINUE
500           CONTINUE

C         If this is primary eigenvalue, do not compute a new d.

              IF (IMODE .EQ. 1) GO TO 3900


C         Compute new dynamical matrix for computation of next higher mode.

C         Normalize eigenvectors such that (PHI transposed)(M)(PHI) = 1.

3000      PROD = 0.0
```

```
      DO 2000 IROW = 1,N
           SUM = 0.0
           DO 2100 ICOL = 1,N
                SUM = SUM + M(IROW,ICOL)*PHI(ICOL)
2100       CONTINUE
           PROD = PROD + PHI(IROW)*SUM
2000  CONTINUE

      PROD = CSQRT(PROD)
      DO 2200 I = 1,N
           PHI(I) = PHI(I)/PROD
2200  CONTINUE

C     D = D + (PHI)(PHI transposed)(M)/IOM2H
C     First compute second term, then add it to D
           DO 3100 IROW = 1,N
                DO 3200 ICOL = 1,N
                     PPTM(IROW,ICOL) = 0.0
                     DO 3300 ITER = 1,N
                          PPTM(IROW,ICOL) = PPTM(IROW,ICOL) +
     1                       PHI(IROW) * PHI(ITER) * M(ITER,ICOL)
3300                 CONTINUE
                     D(IROW,ICOL)= D(IROW,ICOL) +
     1
         PPTM(IROW,ICOL)/(IOM2S+MU)

3200            CONTINUE
3100       CONTINUE

C     *****************************************************
C     *          Compute a new guess for OMEGA.           *
C     *****************************************************

3900  DO 4000 IGUESS3 = 1,IMAX3
C
C          Compute PHI = D*PHIG
C
           DO 4100 I = 1,N
                PHI(I) = 0.0
                DO 4200 ITER = 1,N
                     PHI(I) = PHI(I) + D(I,ITER)*PHIG(ITER)
4200            CONTINUE
4100       CONTINUE
C
C          Normalize on first element. (This is valid for this problem
C          as we are using a simple rod; the first element will never
C          be zero.)  Store the first element of PHI as a "guess" of
C          IOM2H.  IOM2 = IOMEGA**2, where OMEGA is the frequency of
C          the system.

           IOM2H = -1./PHI(1)
```

```
            DO 4300 I =2,N
                 PHI(I) = PHI(I) / PHI(1)
4300        CONTINUE
          PHI(1) = 1.0
C
C           Check to see if the guess is within tolerance level.
C           Get error vector and compute its norm.
            DO 4310 I= 1,N
                 ERR(I) = PHIG(I) - PHI(I)
4310        CONTINUE
C           Find norm of error vector -- this is the radius of the error
C           sphere.
            ERRNORM2 = 0.0
            DO 4320 ITER = 1,N
                 ERRNORM2 = ERRNORM2 + REAL(ERR(ITER))**2 +
     1                           AIMAG(ERR(ITER))**2
4320        CONTINUE
            ERRNORM = SQRT(ERRNORM2)

            write (9,*) 'errnorm for phi',errnorm

            IF (ERRNORM .GT. EPS1) THEN
               DO 4330 I = 1,N
                  PHIG(I) = PHI(I)
4330           CONTINUE

          ELSE
               IF (IMODE .EQ. 2) THEN
                       MU = MU - IOM2H

C     Make sure shift is in right direction -- abs(mu) should be
C     greater than the magnitude of the last iom2, or mu, calculated.
C     If it isn't, add iom2h rather than subtracting it.

                       IF (ABS(MU) .LT. ABS(MUS)) MU = MU + 2*IOM2H

C              Reset IOM2H:
                       IOM2H = 0.0

      write (10,*) 'mu',mu
      write (9,*) 'mu',mu

                                GOTO 11
               ELSE
                       IF (IMODE .EQ. 1) GO TO 4400

                 ENDIF
            ENDIF
4000  CONTINUE
```

```
C     We did not converge on a new estimate.  If IMODE = 2, keep
C     trying.  If this is the primary mode, reset mu.

              WRITE (9,*) 'NEW ESTIMATE NOT FOUND'

              IF (IMODE .EQ. 2) GO TO 20

              MU = MU - IOM2H

C     Make sure shift is in right direction -- abs(mu) should be
C     greater than the magnitude of the last mu calculated.  If it
C     isn't, add iom2h rather than subtracting it.

              IF (ABS(MU) .LT. ABS(MUS)) MU = MU + 2*IOM2H
              write (9,*) 'reset mu',mu,'iom2h =',iom2h

C     Reset IOM2H:
                IOM2H = 0.0

              GO TO 20

C     Need to check if the new guesses of omega and phi are within
C     tolerance.
C          Get error vector and compute its norm.
4400       DO 5310 I= 1,N
                ERR(I) = PHILG(I) - PHI(I)
5310       CONTINUE
C          Find norm of error vector -- this is the radius of the error
C          sphere.
           ERRNORM2 = 0.0
           DO 5320 ITER = 1,N
                ERRNORM2 = ERRNORM2 + REAL(ERR(ITER))**2 +
     1                    AIMAG(ERR(ITER))**2
5320       CONTINUE
           ERRNORM = SQRT(ERRNORM2)


      DO 5330 I = 1,N
           PHILG(I) = PHI(I)
5330  CONTINUE
20    CONTINUE


           WRITE (10,*) 'DID NOT CONVERGE ON BRANCH NO.',IBRANCH,
     1                  ' EIGENVECTOR NO. ',J

C          No sense computing additional eigenvalues, since they
C          depend on this one.  Exit program.

          write (10,*) 'TERMINATING PROGRAM'
```

```
                GO TO 9999

C         Write eigenvalue and eigenvector to output file
5400      MUS = - IOM2
        IOMAS = IOMA
        IOM2S = IOM2

          WRITE (10,*) 'Branch No.',IBRANCH,' (i*OMEGA)**2',J,
     1                 ' = ',IOM2

          WRITE (10,*) 'iomega',IOMEGA
          WRITE (10,*) 'imrio ',imrio

      PTAOP = 0.0
      DO 6000 IROW = 1,N
          SUM = 0.0
          DO 6100 ICOL = 1,N
               SUM = SUM + (K0(IROW,ICOL)+KE(IROW,ICOL))*PHI(ICOL)
6100      CONTINUE
          PTAOP = PTAOP + PHI(IROW)*SUM
6000  CONTINUE

      PTAQP = 0.0
      DO 6010 IROW = 1,N
          SUM = 0.0
          DO 6110 ICOL = 1,N
               SUM = SUM + (K1(IROW,ICOL)+B*KE(IROW,ICOL))*PHI(ICOL)
6110      CONTINUE
          PTAQP = PTAQP + PHI(IROW)*SUM
6010  CONTINUE

      PTMP = 0.0
      DO 6300 IROW = 1,N
          SUM = 0.0
          DO 6400 ICOL = 1,N
               SUM = SUM + M(IROW,ICOL)*PHI(ICOL)
6400      CONTINUE
          PTMP = PTMP + PHI(IROW)*SUM
6300  CONTINUE

      Q = REAL(IQ)
      TM = REAL(2*IM)
      TMQ = TM + Q
      TMM1 = TM - 1.
      IMRIOC =(((TMQ-1)*B*(IMRIO**(2*IM+IQ))+TMM1*(IMRIO**(2*IM)))*PTMP
     1        +(Q-1.)*(IMRIO**(IQ))*PTAQP-PTAOP)
     1        /((TMQ*B*IMRIO**(2*IM+IQ-1)+TM*(IMRIO**(2*IM-1)))*PTMP
     1        +Q*(IMRIO**(IQ-1))*PTAQP)
      WRITE (10,*) 'imrioc',IMRIOC
```

```
            DO 5500 I=1,N
                 WRITE (10,*)  PHI(I)
 5500       CONTINUE

        IF (J .EQ. N) GOTO 11
 12     CONTINUE
 11     CONTINUE
 10     CONTINUE
 9999   STOP
        END
```

Appendix D. *FORTRAN Program for Modified Matrix Iteration Technique*

This computer program computes the first $2mn$ eigenvectors and eigenvalues for a viscoelastically damped rod using the modified matrix iteration techniques presented in Chapter IV. A flowchart for this program is presented in Appendix A. The program uses unformatted READ and WRITE statements, which may produce output different from that shown in Appendix F on computers other than a VAX/VMS. Some of the input parameters are the number of degrees of freedom and the physical characteristics of the rod and pads, including the mass and stiffness matrices.

```
      PROGRAM VDRMI
C
C     FORTRAN Code for viscoelastically damped rod using matrix
C     iteration.
C
C     Set up variables needed in program
C     Declaration statements for variables.
C
      INTEGER FLAG,IQ,IM,IMAX1,IMAX2,IMAX3,N
      REAL A,ALPHA,B,E0,E1,EPS1,EPS2,EPS3,LE,MAG,RHOE
      COMPLEX IOMA,IMRIOC,IOMAL,IMRIO,PROD,PTAOP,PTAQP,PTMP,
     2                                        SUM,IOM2,IOMEGA,
     3        KE(10,10),KV(10,10),M(10,10),
     4        D(10,10),ERR(10),K0(10,10),K1(10,10),
     4        KINV(10,10),KOMEGA(10,10),PHIG(10),PHILG(10),PPTM(10,10),
     4        PSI(10),PSIG(10),U(10,10),UINV(10,10),
     5                                         PHI(10)
C
C     N = Number of Degrees of Freedom
C     IMAX = Maximum number of times to run iteration loop
C     KV = Stiffness matrix for viscoelastic damping material
C     KE = Stiffness matrix for elastic rod
C     M = Mass Matrix of total structure
C     LE = Length of one element
C     A = Cross-sectional area of rod
C     RHO = material density
C     E = Young's modulus for elastic material.
C     E0,E1,B,ALPHA = Parameters of Young's modulus for viscoelastic
C                     material.
C     IOMEGA = i * system frequency
C     IMRIO = Mth root of IOMEGA
C     IMRIOC = check value for IMRIO
C     IOMA = IOMEGA to the ALPHA
C     IOM2 = IOMEGA squared
C     PHI = eigenvector
C     PHIG = guess at an eigenvector
C     PHILG = last guess at an eigenvector
C     PSI = lower mode eigenvector
C     PSIG = guess at lower mode eigenvector
C     D = Dynamical matrix
C     EPS = Tolerance level

C     Open input and output data files.

      OPEN (UNIT=5,FILE='INPUT',STATUS='OLD')
      OPEN (UNIT=9,FILE='DEBUG',STATUS='NEW')
      OPEN (UNIT=10,FILE='OUTPUT',STATUS='NEW')

      write (10,*) 'OUTPUT FOR VDRMI'

C     Set value of PI
```

```
      PI = 3.141592654

C     Read in EPS, RHO, A, LE, parameters for E
C
      READ (5,*) EPS0,EPS1,EPS2,EPS3
      READ (5,*) RHOE
      READ (5,*) AE
      READ (5,*) LE
      READ (5,*) E
      READ (5,*) E0,E1,B

      write (10,*) 'eps0',eps0,'  eps1',eps1
      write (10,*) 'eps2',eps2,' eps3',eps3
      write (10,*) 'rhoe',rhoe
      write (10,*) 'ae',ae
      write (10,*) 'le',le
      write (10,*) 'e',e
      write (10,*) 'e0',e0,'  e1',e1,'  b',b

      READ (5,*) N, IQ, IM
      ALPHA  = REAL(IQ)/REAL(IM)
      READ (5,*) IMAX1,IMAX2,IMAX3

      write (10,*) 'n',n,' iq',iq,' im',im,' alpha',alpha
      write (10,*) 'imax1',imax1,'  imax2',imax2,'  imax3',imax3

C     Read in stiffness and mass matrices for both elastic and
C     viscoelastic materials.

      DO 40 ICOL = 1,N
           DO 50 IROW = 1,N
                READ (5,*) KE(IROW,ICOL)

      write (10,*)'ke',irow,icol,ke(irow,icol)

                 KE(IROW,ICOL) = KE(IROW,ICOL)*E*AE/LE

      write (10,*)'ke',irow,icol,ke(irow,icol)

50         CONTINUE
40    CONTINUE

      DO 42 ICOL = 1,N
           DO 52 IROW = 1,N
                READ (5,*) KV(IROW,ICOL)

      write (10,*)'kv',irow,icol,kv(irow,icol)

C     Compute KO and K1 elements.

                K0(IROW,ICOL)    = E0*KV(IROW,ICOL)
```

```
                   K1(IROW,ICOL)    = E1*KV(IROW,ICOL)

52          CONTINUE
42      CONTINUE

        DO 43 ICOL = 1,N
             DO 53 IROW = 1,N
                  READ (5,*) M(IROW,ICOL)
        write (10,*)'m',irow,icol,m(irow,icol)

                M(IROW,ICOL) = M(IROW,ICOL)*RHOE*AE*LE/6.
        write (10,*)'m',irow,icol,m(irow,icol)

53          CONTINUE
43      CONTINUE

C***************************************************************
C*           Calculate roots for branch = IBRANCH              *
C***************************************************************

        DO 10 IBRANCH = 0,2*IM-1

C            Set initial value of iom2

             IOM2 = CMPLX(1.0,1.0)

C       Reset FLAG

             FLAG = 0

             DO 11 J = 1,N

                   WRITE(9,*) 'Computing Eigenvalue No.',j,'on
branch',ibranch
           print *, 'Computing Eigenvalue No.',j,'on branch',ibranch

        DO 20 IGUESS1 = 1,IMAX1

C       Check if this is the first iteration on same branch -- if so, use
C       current value of ioma and skip right to computation of new D.

        IF(FLAG .EQ. 1) GOTO 900

C       Calculate IOMA = (i*omega)**alpha = ((i*omega)**2)**alpha/2
C                  = (iom2)**alpha/2
C                = (abs(iom2)*exp(i*ang))**alpha/2
C                = ((mag**(1/im))*exp(i*ang*(1/2*im))**iq = IMRIO ** iq

             REIOM2 = REAL(IOM2)
             AIMIOM2  = AIMAG(IOM2)
```

```
          MAG    = SQRT(REIOM2**2 + AIMIOM2**2)
          ANG    = ATAN2(AIMIOM2,REIOM2)

      IF (ANG .LT. 0) ANG = ANG + 2*PI

          ARG = (ANG + 2*IBRANCH*PI)/(2*IM)

          IMRIO = (MAG**(1/REAL(2*IM)))*CMPLX(COS(ARG),SIN(ARG))

        IOMEGA = (IMRIO**IM)

          IOMA = IMRIO**IQ

C     Now compute K(omega).

29    DO 30 ICOL = 1,N
          DO 31 IROW = 1,N
            KOMEGA(IROW,ICOL) = KE(IROW,ICOL) +
     1      (KC(IROW,ICOL) + IOMA*K1(IROW,ICOL))/(1 + B*IOMA)
31        CONTINUE
30    CONTINUE

          DIF = ABS(IOMA - IOMAL)

          write (9,*) 'dif',dif,'errnorm',errnorm

          IF (DIF .LT. EPS0 .AND. ERRNORM .LT. EPS1) GO TO 5400
          IOMAL = IOMA


C     **********************************************************
C     *                                                        *
C     *              Compute dynamical matrix                  *
C     *                                                        *
C     **********************************************************
C
C         First compute inverse of K by using a Cholesky decomposition
C              KOMEGA = Utranspose * U
C         Compute U
          U(1,1) = CSQRT(KOMEGA(1,1))

          DO 110 ICOL = 2,N
               U(1,ICOL) = KOMEGA(1,ICOL)/U(1,1)
110       CONTINUE

          DO 120 IROW = 2,N
               SUM = 0.0
               DO 130 ITER = 1,IROW-1
                    SUM = SUM + U(ITER,IROW)**2
130            CONTINUE
               U(IROW,IROW) = CSQRT(KOMEGA(IROW,IROW)-SUM)
```

```
               DO 140 ICOL = IROW+1,N
                  SUM = 0.0
                  DO 150 ITER =1,IROW-1
                     SUM = SUM + U(ITER,IROW)*U(ITER,ICOL)
150               CONTINUE
                  U(IROW,ICOL) = (KOMEGA(IROW,ICOL)-SUM)/U(IROW,IROW)

140            CONTINUE
120         CONTINUE
C           inverse of KOMEGA = (inverse of U)*(inverse of U,transposed)
C              First calculate inverse of U
               DO 200 ICOL = 1,N
                  UINV(ICOL,ICOL) = 1.0/U(ICOL,ICOL)

                  DO 210 IROW = 1,ICOL-1
                     SUM = 0.0
                     DO 220 ITER = IROW, ICOL-1
                        SUM = SUM + UINV(IROW,ITER)*U(ITER,ICOL)
220                  CONTINUE
                     UINV(IROW,ICOL) = -SUM/U(ICOL,ICOL)

210               CONTINUE
200            CONTINUE
C              Now for KOMEGA inverse
               DO 230 IROW = 1,N
                  DO 240 ICOL =1,N
                  KINV(IROW,ICOL) = 0.0
                     DO 250 ITER = 1,N
                        KINV(IROW,ICOL) = KINV(IROW,ICOL) +
     1                   UINV(IROW,ITER) * UINV(ICOL,ITER)
250                  CONTINUE
240               CONTINUE
230            CONTINUE
C           Compute D = KINV * M
            DO 500 IROW = 1,N
               DO 600 ICOL = 1,N
                  D(IROW,ICOL) = 0.0
                  DO 700 ITER = 1,N
                   D(IROW,ICOL) = D(IROW,ICOL) +
     1                              KINV(IROW,ITER) * M(ITER,ICOL)
700               CONTINUE
600            CONTINUE
500         CONTINUE

C     If J > 1, compute lower order modes and calculate a new
C      dynamical matrix, one with the lower modes subtracted off.

            IF (J .EQ. 1) GO TO 3900

C     **********************************************************
```

```
C     *    Compute lower modes                                     *
C     ****************************************************************
C
900   DO 1000 IMODE = 1, J-1

      IF(FLAG .EQ. 1) THEN
           FLAG = 0
           GOTO 3000
      ENDIF

C          Set initial eigenvector guess.
           DO 1010 I = 1,N
                PSIG(I) = 1.0/I
1010       CONTINUE
      DO 1050 IGUESS2 = 1,IMAX2
C
C          Compute PSI = D*PSIG
C
           DO 1100 I = 1,N
                PSI(I) = 0.0
                DO 1200 ITER = 1,N
                     PSI(I) = PSI(I) + D(I,ITER)*PSIG(ITER)
1200            CONTINUE


1100       CONTINUE
C
C          Normalize on first element. (This is valid for this problem
C          as we are using a simple rod; the first element will never
C          be zero.)  Store the first element as a "guess" of IOM2.
C          IOM2 = (I*W)**2, where W is the frequency of the system.

           IOM2 = -1./PSI(1)

           DO 1300 I =2,N
                PSI(I) = PSI(I) / PSI(1)
1300       CONTINUE
           PSI(1) = 1.0
C
C          Check to see if the guess is within tolerance level.
C          Get error vector and compute its norm.
           DO 1310 I= 1,N
                ERR(I) = PSIG(I) - PSI(I)
1310       CONTINUE
C          Find norm of error vector -- this is the radius of the error
C          sphere.
           ERRNORM2 = 0.0
           DO 1320 ITER = 1,N
                ERRNORM2 = ERRNORM2 + REAL(ERR(ITER))**2 +
     1                         AIMAG(ERR(ITER))**2
1320       CONTINUE
```

```
            ERRNORM = SQRT(ERRNORM2)

      write(9,*) 'errnorm for psi',errnorm

            IF (ERRNORM .LT. EPS2) GO TO 3000
      DO 1330 I = 1,N
            PSIG(I) = PSI(I)
1330  CONTINUE
1050  CONTINUE
      WRITE (10,*) 'DID NOT CONVERGE ON MODE',IMODE,
     1             ' EIGENVECTOR NO.',J,'BRANCH NO.',IBRANCH
      WRITE (10,*) 'TERMINATING PROGRAM'

                  GO TO 9999

C     Compute new dynamical matrix for computation of next eigenvalue
C     and eigenvector.

C     Normalize eigenvectors such that (PSI transposed)(M)(PSI) = 1.

3000  PROD = 0.0
      DO 2000 IROW = 1,N
            SUM = 0.0
            DO 2100 ICOL = 1,N
                  SUM = SUM + M(IROW,ICOL)*PSI(ICOL)
2100        CONTINUE
            PROD = PROD + PSI(IROW)*SUM
2000  CONTINUE

      PROD = CSQRT(PROD)
      DO 2200 I = 1,N
            PSI(I) = PSI(I)/PROD
2200  CONTINUE

C     D = D + (1./IOM2)(PSI)(PSI transposed)(M)
C     First compute second term, then subtract it from D
            DO 3100 IROW = 1,N
                  DO 3200 ICOL = 1,N
                        PPTM(IROW,ICOL) = 0.0
                        DO 3300 ITER = 1,N
                              PPTM(IROW,ICOL) = PPTM(IROW,ICOL) +
     1                           PSI(IROW) * PSI(ITER) * M(ITER,ICOL)
3300                    CONTINUE
                        D(IROW,ICOL) = D(IROW,ICOL) + 1./IOM2 *
     1                                 PPTM(IROW,ICOL)

3200              CONTINUE
3100        CONTINUE
1000  CONTINUE

C     *****************************************************
```

```
C     *           Compute a new guess for OMEGA.          *
C     ****************************************************

3900  DO 4000 IGUESS3 = 1,IMAX3

C        Check if this is the first iteration on this eigenvalue.

         IF (IGUESS1 .GT. 2 .OR. IGUESS3 .GT. 1) GOTO 3950

C          Set initial eigenvector guess.

           DO 15 I = 1,N
                PHIG(I) = 1.0/I
15         CONTINUE

C          Compute PHI = D*PHIG
C
3950       DO 4100 I = 1,N
                PHI(I) = 0.0
                DO 4200 ITER = 1,N
                     PHI(I) = PHI(I) + D(I,ITER)*PHIG(ITER)
4200            CONTINUE
4100       CONTINUE
C
C          Normalize on first element. (This is valid for this problem
C          as we are using a simple rod; the first element will never
C          be zero.)  Store the first element as a "guess" of IOM1.
C          IOM2 = (IOMEGA**2), where IOMEGA is  i times the frequency
of the
C          system.

           IOM2 = -1./PHI(1)

           DO 4300 I =2,N
                PHI(I) = PHI(I) / PHI(1)
4300       CONTINUE
         PHI(1) = 1.0
C
C          Check to see if the guess is within tolerance level.
C          Get error vector and compute its norm.
           DO 4310 I= 1,N
                ERR(I) = PHIG(I) - PHI(I)
4310       CONTINUE
C          Find norm of error vector -- this is the radius of the error
C          sphere.
           ERRNORM2 = 0.0
           DO 4320 ITER = 1,N
                ERRNORM2 = ERRNORM2 + REAL(ERR(ITER))**2 +
     1                          AIMAG(ERR(ITER))**2
4320       CONTINUE
           ERRNORM = SQRT(ERRNORM2)
```

```
          write (9,*) 'errnorm for phi',errnorm

          IF (ERRNORM .LT. EPS3) GO TO 4400
      DO 4330 I = 1,N
          PHIG(I) = PHI(I)
4330  CONTINUE
4000  CONTINUE

          WRITE (10,*) 'DID NOT CONVERGE ON BRANCH NO.',IBRANCH,
     1                   ' EIGENVECTOR NO. ',J

C         No sense computing additional eigenvalues, since they
C         depend on this one.  Exit program.

         write (10,*) 'TERMINATING PROGRAM'

                  GO TO 9999

C     Need to check if the new guesses of omega and phi are within
C     tolerance.
C         Get error vector and compute its norm.
4400      DO 5310 I= 1,N
              ERR(I) = PHILG(I) - PHI(I)
5310      CONTINUE
C         Find norm of error vector -- this is the radius of the error
C         sphere.
          ERRNORM2 = 0.0
          DO 5320 ITER = 1,N
              ERRNORM2 = ERRNORM2 + REAL(ERR(ITER))**2 +
     1                       AIMAG(ERR(ITER))**2
5320      CONTINUE
          ERRNORM = SQRT(ERRNORM2)


      DO 5330 I = 1,N
          PHILG(I) = PHI(I)
5330  CONTINUE
20    CONTINUE

          WRITE (10,*) 'DID NOT CONVERGE ON BRANCH NO.',IBRANCH,
     1                   ' EIGENVECTOR NO. ',J

C         No sense computing additional eigenvalues, since they
C         depend on this one.  Exit program.

         write (10,*) 'TERMINATING PROGRAM'

                  GO TO 9999

C         Write eigenvalue and eigenvector to output file
```

```
5400       WRITE (10,*) 'Branch No.',IBRANCH,' Eigenvalue No.',J,
     1                  ' = ',IOM2
           WRITE (10,*) 'iomega',IOMEGA
           WRITE (10,*) 'imrio ',imrio

      PTAOP = 0.0
      DO 6000 IROW = 1,N
           SUM = 0.0
           DO 6100 ICOL = 1,N
                SUM = SUM + (K0(IROW,ICOL)+KE(IROW,ICOL))*PHI(ICOL)
6100       CONTINUE
           PTAOP = PTAOP + PHI(IROW)*SUM
6000  CONTINUE

      PTAQP = 0.0
      DO 6010 IROW = 1,N
           SUM = 0.0
           DO 6110 ICOL = 1,N
                SUM = SUM + (K1(IROW,ICOL)+B*KE(IROW,ICOL))*PHI(ICOL)
6110       CONTINUE
           PTAQP = PTAQP + PHI(IROW)*SUM
6010  CONTINUE

      PTMP = 0.0
      DO 6300 IROW = 1,N
           SUM = 0.0
           DO 6400 ICOL = 1,N
                SUM = SUM + M(IROW,ICOL)*PHI(ICOL)
6400       CONTINUE
           PTMP = PTMP + PHI(IROW)*SUM
6300  CONTINUE

      Q = REAL(IQ)
      TM = REAL(2*IM)
      TMQ = TM + Q
      TMM1 = TM - 1.
      IMRIOC =(((TMQ-1)*B*(IMRIO**(2*IM+IQ))+TMM1*(IMRIO**(2*IM)))*PTMP
     1        +(Q-1.)*(IMRIO**(IQ))*PTAQP-PTAOP)
     1        /((TMQ*B*IMRIO**(2*IM+IQ-1)+TM*(IMRIO**(2*IM-1)))*PTMP
     1        +Q*(IMRIO**(IQ-1))*PTAQP)
      WRITE (10,*) 'imrioc',IMRIOC


           DO 5500 I=1,N
                WRITE (10,*)  PHI(I)

C               Set PSI = PHI for first computation of new D.
                PSI(I) = PHI(I)
5500       CONTINUE
```

```
C     Let the first guess of iomega be the last value.  Set FLAG.

      FLAG = 1

11    CONTINUE
10    CONTINUE
9999  STOP
      END
```

Appendix E. *FORTRAN Program for Eigenvalues Due to Nonzero B*

This computer program computes the additional nq eigenvectors and eigenvalues for a viscoelastically damped rod that arise due to a nonzero b in the viscoelastic model (see Equation 28). The techniques techniques presented in Chapter IV are used in this program. A flowchart for this program is presented in Appendix A. The program uses unformatted READ and WRITE statements, which may produce output different from that shown in Appendix F on computers other than a VAX/VMS. Some of the input parameters are the number of degrees of freedom and the physical characteristics of the rod and pads, including the mass and stiffness matrices.

```
      PROGRAM NONZEROB
C
C     FORTRAN Code to compute structural modes due to non-zero b
C     for viscoelastically damped rod using matrix iteration.
C
C     Set up variables needed in program
C     Declaration statements for variables.
C
      INTEGER FLAG,IQ,IM,IMAX1,IMAX2,IMAX3,N
      REAL A,ALPHA,B,E0,E1,EPS1,EPS2,EPS3,LE,MAG,RHOE
      COMPLEX BIOMAP2,IOMA,IMRIOC,IOMAL,IMRIO,IMRIOL,
     1        PROD,PTAOP,PTAQP,PTMP,
     2                                        SUM,IOM2,IOMEGA,
     3        KE(10,10),KV(10,10),M(10,10),
     4        D(10,10),ERR(10),K0(10,10),K1(10,10),
     4        KINV(10,10),KOMEGA(10,10),PHIG(10),PHILG(10),PPTM(10,10),
     4        PSI(10),PSIG(10),U(10,10),UINV(10,10),
     5                                          PHI(10)
C
C     N = Number of Degrees of Freedom
C     IMAX = Maximum number of times to run iteration loop
C     KV = Stiffness matrix for viscoelastic damping material
C     KE = Stiffness matrix for elastic rod
C     M = Mass Matrix of total structure
C     LE = Length of one element
C     A = Cross-sectional area of rod
C     RHO = material density
C     E = Young's modulus for elastic material.
C     E0,E1,B,ALPHA = Parameters of Young's modulus for viscoelastic
C                     material.
C     IOMEGA = i * system frequency
C     IMRIO = Mth root of IOMEGA
C     IMRIOC = check value for IMRIO
C     IOMA = IOMEGA to the ALPHA
C     IOM2 = IOMEGA squared
C     PHI = eigenvector
C     PHIG = guess at an eigenvector
C     PHILG = last guess at an eigenvector
C     PSI = pseudoeigenvector
C     PSIG = guess at psuedoeigenvector
C     D = Dynamical matrix
C     EPS = Tolerance level

C     Open input and output data files.

      OPEN (UNIT=5,FILE='INPUT',STATUS='OLD')
      OPEN (UNIT=9,FILE='DEBUG',STATUS='NEW')
      OPEN (UNIT=10,FILE='OUTPUT',STATUS='NEW')

      write (10,*) 'OUTPUT FOR NONZEROB'
```

```
C     Set value of PI
      PI = 3.141592654

C     Read in EPS, RHO, A, LE, parameters for E
C
      READ (5,*) EPS0,EPS1,EPS2,EPS3
      READ (5,*) RHOE
      READ (5,*) AE
      READ (5,*) LE
      READ (5,*) E
      READ (5,*) E0,E1,B

      write (10,*) 'eps0',eps0,'  eps1',eps1
      write (10,*) 'eps2',eps2,' eps3',eps3
      write (10,*) 'rhoe',rhoe
      write (10,*) 'ae',ae
      write (10,*) 'le',le
      write (10,*) 'e',e
      write (10,*) 'e0',e0,'  e1',e1,'  b',b

      READ (5,*) N, IQ, IM
      ALPHA  = REAL(IQ)/REAL(IM)
      READ (5,*) IMAX1,IMAX2,IMAX3

      write (10,*) 'n',n,' iq',iq,' im',im,' alpha',alpha
      write (10,*) 'imax1',imax1,'  imax2',imax2,'  imax3',imax3

C     Read in stiffness and mass matrices for both elastic and
C     viscoelastic materials.

      DO 40 ICOL = 1,N
           DO 50 IROW = 1,N
                READ (5,*) KE(IROW,ICOL)

      write (10,*)'ke',irow,icol,ke(irow,icol)

                 KE(IROW,ICOL) = KE(IROW,ICOL)*E*AE/LE

      write (10,*)'ke',irow,icol,ke(irow,icol)

50         CONTINUE
40    CONTINUE

      DO 42 ICOL = 1,N
           DO 52 IROW = 1,N
                READ (5,*) KV(IROW,ICOL)

      write (10,*)'kv',irow,icol,kv(irow,icol)

C     Compute K0 and K1 elements.
```

```
                  K0(IROW,ICOL)    = E0*KV(IROW,ICOL)
                  K1(IROW,ICOL)    = E1*KV(IROW,ICOL)


52            CONTINUE
42       CONTINUE

         DO 43 ICOL = 1,N
              DO 53 IROW = 1,N
                   READ (5,*) M(IROW,ICOL)
         write (10,*)'m',irow,icol,m(irow,icol)

                   M(IROW,ICOL) = M(IROW,ICOL)*RHOE*AE*LE/6.
         write (10,*)'m',irow,icol,m(irow,icol)

53            CONTINUE
43       CONTINUE

C****************************************************************
C*            Calculate roots for branch = IBRANCH              *
C****************************************************************

         DO 10 IBRANCH = 0,IQ-1

C             Set initial value of iom2

              IMRIO = CMPLX(-1.0/B)
              IOM2 = IMRIO**(2*IM)
              BIOMAP2 = 0.0

C        Reset FLAG

              FLAG = 0

              DO 11 J = 1,N

              WRITE(9,*) 'Computing Eigenvalue No.',j,'on branch',ibranch
              print *, 'Computing Eigenvalue No.',j,'on branch',ibranch

         DO 20 IGUESS1 = 1,IMAX1

C        Check if this is the first iteration on same branch -- if so, use
C        current value of ioma and skip right to computation of new D.

         IF(FLAG .EQ. 1) GOTO 900

C        Calculate IOMA = (i*omega)**alpha = IMRIO ** iq
C                  IMRIO = (Q - iom2)/(b*iom2*imrio**(q-1))
C                      Q = b*(i*omega)**(alpha+2) + iom2 = biomap2

              IOMA = ((BIOMAP2 - IOM2)/(B*(IMRIO**(2*IM+IQ-1))))**IQ
```

```
          REIOMA = REAL(IOMA)
          AIMIOMA  = AIMAG(IOMA)
          MAG    = SQRT(REIOMA**2 + AIMIOMA**2)
          ANG    = ATAN2(AIMIOMA,REIOMA)

        IF (ANG .LT. 0) ANG = ANG + 2*PI

          ARG = (ANG + 2*IBRANCH*PI)/REAL(IQ)

          IMRIO = (MAG**(1/REAL(IQ)))*CMPLX(COS(ARG),SIN(ARG))

          IOMEGA = IMRIO**IM

          IOMA = IMRIO**IQ
          IOM2 = IOMEGA**2

        print *,'biomap2',biomap2,'iomA',iomA

C     Now compute K(omega).

29    DO 30 ICOL = 1,N
          DO 31 IROW = 1,N
            KOMEGA(IROW,ICOL) = (1+B*IOMA)*KE(IROW,ICOL) +
     1                          K0(IROW,ICOL) + IOMA*K1(IROW,ICOL)
31        CONTINUE
30    CONTINUE

          DIF = ABS(IMRIO - IMRIOL)

          write (9,*) 'dif',dif,'errnorm',errnorm

          IF (DIF .LT. EPS0 .AND. ERRNORM .LT. EPS1) GO TO 5400
          IMRIOL = IMRIO


C     *********************************************************
C     *                                                       *
C     *              Compute dynamical matrix                 *
C     *                                                       *
C     *********************************************************
C
C         First compute inverse of K by using a Cholesky decomposition
C              KOMEGA = Utranspose * U
C         Compute U
          U(1,1) = CSQRT(KOMEGA(1,1))

          DO 110 ICOL = 2,N
               U(1,ICOL) = KOMEGA(1,ICOL)/U(1,1)
110       CONTINUE

          DO 120 IROW = 2,N
```

```
                SUM = 0.0
                DO 130 ITER = 1,IROW-1
                     SUM = SUM + U(ITER,IROW)**2
130             CONTINUE
                U(IROW,IROW) = CSQRT(KOMEGA(IROW,IROW)-SUM)

                DO 140 ICOL = IROW+1,N
                    SUM = 0.0
                    DO 150 ITER =1,IROW-1
                         SUM = SUM + U(ITER,IROW)*U(ITER,ICOL)
150                 CONTINUE
                    U(IROW,ICOL) = (KOMEGA(IROW,ICOL)-SUM)/U(IROW,IROW)

140             CONTINUE
120        CONTINUE
C          inverse of KOMEGA = (inverse of U)*(inverse of U,transposed)
C               First calculate inverse of U
                DO 200 ICOL = 1,N
                     UINV(ICOL,ICOL) = 1.0/U(ICOL,ICOL)

                     DO 210 IROW = 1,ICOL-1
                          SUM = 0.0
                          DO 220 ITER = IROW, ICOL-1
                               SUM = SUM + UINV(IROW,ITER)*U(ITER,ICOL)
220                       CONTINUE
                          UINV(IROW,ICOL) = -SUM/U(ICOL,ICOL)

210                  CONTINUE
200             CONTINUE
C               Now for KOMEGA inverse
                DO 230 IROW = 1,N
                     DO 240 ICOL =1,N
                     KINV(IROW,ICOL) = 0.0
                          DO 250 ITER = 1,N
                              KINV(IROW,ICOL) = KINV(IROW,ICOL) +
    1                          UINV(IROW,ITER) * UINV(ICOL,ITER)
250                       CONTINUE
240                  CONTINUE
230             CONTINUE
C          Compute D = KINV * M
           DO 500 IROW = 1,N
                DO 600 ICOL = 1,N
                   D(IROW,ICOL) = 0.0
                   DO 700 ITER = 1,N
                     D(IROW,ICOL) = D(IROW,ICOL) +
    1                               KINV(IROW,ITER) * M(ITER,ICOL)
700                CONTINUE
600             CONTINUE
500        CONTINUE

C     If J > 1, compute lower order modes and calculate a new
```

```
C      dynamical matrix, one with the lower modes subtracted off.

            IF (J .EQ. 1) GO TO 3900

C      ******************************************************
C      *    Compute lower modes                             *
C      ******************************************************
C
900    DO 1000 IMODE = 1, J-1

       IF(FLAG .EQ. 1) THEN
            FLAG = 0
            GOTO 3000
       ENDIF

C           Set initial eigenvector guess.
            DO 1010 I = 1,N
                 PSIG(I) = 1.0/I
1010        CONTINUE
       DO 1050 IGUESS2 = 1,IMAX2
C
C           Compute PSI = D*PSIG
C
            DO 1100 I = 1,N
                 PSI(I) = 0.0
                 DO 1200 ITER = 1,N
                      PSI(I) = PSI(I) + D(I,ITER)*PSIG(ITER)
1200             CONTINUE


1100        CONTINUE
C
C           Normalize on first element. (This is valid for this problem
C           as we are using a simple rod; the first element will never
C           be zero.)  Store the first element as a "guess" of BIOMAP2.
C           BIOMAP2 = B*(I*W)**(ALPHA+2)+IOM2, where W is the frequency
C           of the system.

            BIOMAP2 = -1./PSI(1)

            DO 1300 I =2,N
                 PSI(I) = PSI(I) / PSI(1)
1300        CONTINUE
            PSI(1) = 1.0
C
C           Check to see if the guess is within tolerance level.
C           Get error vector and compute its norm.
            DO 1310 I= 1,N
                 ERR(I) = PSIG(I) - PSI(I)
1310        CONTINUE
C           Find norm of error vector -- this is the radius of the error
```

```
C          sphere.
           ERRNORM2 = 0.0
           DO 1320 ITER = 1,N
                ERRNORM2 = ERRNORM2 + REAL(ERR(ITER))**2 +
     1                    AIMAG(ERR(ITER))**2
1320       CONTINUE
           ERRNORM = SQRT(ERRNORM2)


           IF (ERRNORM .LT. EPS2) GO TO 3000
      DO 1330 I = 1,N
           PSIG(I) = PSI(I)
1330  CONTINUE
1050  CONTINUE
      WRITE (10,*) 'DID NOT CONVERGE ON MODE',IMODE,
     1             ' EIGENVECTOR NO.',J,'BRANCH NO.',IBRANCH
      WRITE (10,*) 'TERMINATING PROGRAM'

                 GO TO 9999

C     Compute new dynamical matrix for computation of next eigenvalue
C     and eigenvector.

C     Normalize eigenvectors such that (PSI transposed)(M)(PSI) = 1.

3000  PROD = 0.0
      DO 2000 IROW = 1,N
           SUM = 0.0
           DO 2100 ICOL = 1,N
                SUM = SUM + M(IROW,ICOL)*PSI(ICOL)
2100       CONTINUE
           PROD = PROD + PSI(IROW)*SUM
2000  CONTINUE

      PROD = CSQRT(PROD)
      DO 2200 I = 1,N
           PSI(I) = PSI(I)/PROD
2200  CONTINUE

C   D = D + (1/BIOMAP2)(PSI)(PSI transposed)(M)
C     First compute second term, then subtract it from D
           DO 3100 IROW = 1,N
                DO 3200 ICOL = 1,N
                     PPTM(IROW,ICOL) = 0.0
                     DO 3300 ITER = 1,N
                          PPTM(IROW,ICOL) = PPTM(IROW,ICOL) +
     1                       PSI(IROW) * PSI(ITER) * M(ITER,ICOL)
3300                 CONTINUE
                     D(IROW,ICOL) = D(IROW,ICOL) + 1./BIOMAP2 *
     1                              PPTM(IROW,ICOL)
```

```
3200            CONTINUE
3100        CONTINUE
1000    CONTINUE

C       ****************************************************
C       *           Compute a new guess for BIOMAP2.       *
C       ****************************************************

3900    DO 4000 IGUESS3 = 1,IMAX3

C           Check if this is the first iteration on this eigenvalue.

            IF (IGUESS1 .GT. 2 .OR. IGUESS3 .GT. 1) GOTO 3950

C           Set initial eigenvector guess.

            DO 15 I = 1,N
                PHIG(I) = 1.0/I
15          CONTINUE

C           Compute PHI = D*PHIG
C
3950        DO 4100 I = 1,N
                PHI(I) = 0.0
                DO 4200 ITER = 1,N
                    PHI(I) = PHI(I) + D(I,ITER)*PHIG(ITER)
4200            CONTINUE
4100        CONTINUE
C
C           Normalize on first element. (This is valid for this problem
C           as we are using a simple rod; the first element will never
C           be zero.)  Store the first element as a "guess" of BIOMAP2.
C           BIOMAP2 = B*(IOMEGA**2) + IOM2, where IOMEGA is  i times the

C           frequency of the system.

            BIOMAP2 = -1./PHI(1)

            DO 4300 I =2,N
                PHI(I) = PHI(I) / PHI(1)
4300        CONTINUE
            PHI(1) = 1.0
C
C           Check to see if the guess is within tolerance level.
C           Get error vector and compute its norm.
            DO 4310 I= 1,N
                ERR(I) = PHIG(I) - PHI(I)
4310        CONTINUE
C           Find norm of error vector -- this is the radius of the error
C           sphere.
            ERRNORM2 = 0.0
```

```
          DO 4320 ITER = 1,N
               ERRNORM2 = ERRNORM2 + REAL(ERR(ITER))**2 +
     1                   AIMAG(ERR(ITER))**2
4320      CONTINUE
          ERRNORM = SQRT(ERRNORM2)

          write (9,*) 'errnorm for phi',errnorm

          IF (ERRNORM .LT. EPS3) GO TO 4400
     DO 4330 I = 1,N
          PHIG(I) = PHI(I)
4330 CONTINUE
4000 CONTINUE

          WRITE (10,*) 'DID NOT CONVERGE ON BRANCH NO.',IBRANCH,
     1                 ' EIGENVECTOR NO. ',J

C         No sense computing additional eigenvalues, since they
C         depend on this one.  Exit program.

         write (10,*) 'TERMINATING PROGRAM'

                 GO TO 9999

C    Need to check if the new guesses of omega and phi are within
C    tolerance.
C         Get error vector and compute its norm.
4400      DO 5310 I= 1,N
               ERR(I) = PHILG(I) - PHI(I)
5310      CONTINUE
C         Find norm of error vector -- this is the radius of the error
C         sphere.
          ERRNORM2 = 0.0
          DO 5320 ITER = 1,N
               ERRNORM2 = ERRNORM2 + REAL(ERR(ITER))**2 +
     1                   AIMAG(ERR(ITER))**2
5320      CONTINUE
          ERRNORM = SQRT(ERRNORM2)


     DO 5330 I = 1,N
          PHILG(I) = PHI(I)
5330 CONTINUE
20   CONTINUE

          WRITE (10,*) 'DID NOT CONVERGE ON BRANCH NO.',IBRANCH,
     1                 ' EIGENVECTOR NO. ',J

C         No sense computing additional eigenvalues, since they
C         depend on this one.  Exit program.
```

```
          write (10,*) 'TERMINATING PROGRAM'

                   GO TO 9999

C        Write eigenvalue and eigenvector to output file
5400     WRITE (10,*) 'Branch No.',IBRANCH,' Eigenvalue No.',J,
     1                ' = ',IOM2
         WRITE (10,*) 'iomega',IOMEGA
         WRITE (10,*) 'imrio',imrio


      PTAOP = 0.0
      DO 6000 IROW = 1,N
         SUM = 0.0
         DO 6100 ICOL = 1,N
            SUM = SUM + (K0(IROW,ICOL)+KE(IROW,ICOL))*PHI(ICOL)
6100     CONTINUE
         PTAOP = PTAOP + PHI(IROW)*SUM
6000  CONTINUE

      PTAQP = 0.0
      DO 6010 IROW = 1,N
         SUM = 0.0
         DO 6110 ICOL = 1,N
            SUM = SUM + (K1(IROW,ICOL)+B*KE(IROW,ICOL))*PHI(ICOL)
6110     CONTINUE
         PTAQP = PTAQP + PHI(IROW)*SUM
6010  CONTINUE

      PTMP = 0.0
      DO 6300 IROW = 1,N
         SUM = 0.0
         DO 6400 ICOL = 1,N
            SUM = SUM + M(IROW,ICOL)*PHI(ICOL)
6400     CONTINUE
         PTMP = PTMP + PHI(IROW)*SUM
6300  CONTINUE

      Q = REAL(IQ)
      TM = REAL(2*IM)
      TMQ = TM + Q
      TMM1 = TM - 1.
      IMRIOC =(((TMQ-1)*B*(IMRIO**(2*IM+IQ))+TMM1*(IMRIO**(2*IM)))*PTMP
     1        +(Q-1.)*(IMRIO**(IQ))*PTAQP-PTAOP)
     1        /((TMQ*B*IMRIO**(2*IM+IQ-1)+TM*(IMRIO**(2*IM-1)))*PTMP
     1        +Q*(IMRIO**(IQ-1))*PTAQP)

      WRITE (10,*) 'imrioc',IMRIOC

         DO 5500 I=1,N
            WRITE (10,*)  PHI(I)
```

```
C                Set PSI = PHI for first computation of new D.
                 PSI(I) = PHI(I)
5500        CONTINUE

C     Let the first guess of iomega be the last value.  Set FLAG.

      FLAG = 1

11    CONTINUE
10    CONTINUE
9999  STOP
      END
```

Appendix F. *Sample Eigenstructure for Ten-by-Ten System*

The eigenstructure for the sample problem discussed in Chapter VI is included here for completeness. The output is in the form that the programs printed it out, and all $n(2m + q)$ eigenvalues are listed.

```
OUTPUT FOR VDRSS1
rhoe   2710.000
ae  6.2500000E-02
le  0.9090000
e  5.5160001E+10
e0   4750000.      el   1843750.      b  1.0000000E-03
n           10 iq           1 im           2 alpha  0.5000000
imax1          40  imax2          140  imax3         200
eps0  1.0000000E-07  eps1  9.9999993E-04
Branch No.  0 (i*OMEGA)**2                1 =  (-2380414.,-330270.6)
iomega (-106.7766,1546.550)
imrio  (26.86498,28.78374)
imrioc (26.86460,28.78333)
'1.000000,0.0000000E+00)
(1.918981,-4.6963174E-07)
(2.682489,-1.8393910E-06)
(3.228670,-3.8353260E-06)
(3.513276,-6.1834844E-06)
(3.513254,-8.3750992E-06)
(3.228611,-9.6274507E-06)
(2.682413,-9.3926346E-06)
(1.918911,-7.5336757E-06)
(0.9999585,-4.1679818E-06)
mu (8630993.,344225.4)
Branch No.  0 (i*OMEGA)**2                2 =  (-8769234.,-455740.9)
iomega (-76.92407,2962.288)
imrio  (37.98923,38.98853)
imrioc (37.98922,38.98852)
(1.000000,0.0000000E+00)
(1.682507,1.0944797E-07)
(1.830830,1.0944797E-07)
(1.397877,3.2834393E-07)
(0.5211071,5.7460187E-07)
(-0.5211105,7.3877385E-07)
(-1.397880,8.7558379E-07)
(-1.830833,1.0397558E-06)
(-1.682509,7.6613583E-07)
(-1.000001,3.8306791E-07)
mu (1.4748486E+07,250720.7)
Branch No.  0 (i*OMEGA)**2                3 =  (-1.9881194E+07,-584557.6)
iomega (-65.54370,4459.315)
imrio  (46.87352,47.56753)
imrioc (46.87298,47.56708)
(1.000000,0.0000000E+00)
(1.309692,2.1120691E-06)
(0.7152848,6.6443340E-06)
(-0.3729208,1.0633175E-05)
(-1.203747,1.0431492E-05)
(-1.203667,4.1905241E-06)
(-0.3727068,-6.0308817E-06)
(0.7155645,-1.5249475E-05)
```

```
(1.309949,-1.7960991E-05)
(1.000152,-1.2022547E-05)
mu (2.0617300E+07,-44966.94)
Branch No.  0 (i*OMEGA)**2             4 =   (-3.6620676E+07,-734676.0)
iomega (-60.69946,6051.806)
imrio  (54.73304,55.28476)
imrioc (54.73180,55.28397)
(1.000000,0.0000000E+00)
(0.8307522,1.6252874E-05)
(-0.3098879,3.5149289E-05)
(-1.088272,2.4561141E-05)
(-0.5942521,-2.0840604E-05)
(0.5946429,-6.1570696E-05)
(1.088393,-5.0073399E-05)
(0.3096554,1.3819139E-05)
(-0.8311772,7.3235838E-05)
(-1.000324,6.8564186E-05)
mu (4.2753184E+07,-1.9644188E-07)
Branch No.  0 (i*OMEGA)**2             5 =   (-6.0250020E+07,-924936.5)
iomega (-59.57861,7762.316)
imrio  (62.06032,62.53848)
imrioc (62.05998,62.53775)
(1.000000,0.0000000E+00)
(0.2845821,-1.7432709E-05)
(-0.9190395,-1.9537525E-05)
(-0.5461531,1.7321930E-05)
(0.7636402,4.3647233E-05)
(0.7635363,5.6598396E-06)
(-0.5463436,-5.2509618E-05)
(-0.9190938,-3.9639021E-05)
(0.2847274,3.5791942E-05)
(1.000175,6.4373125E-05)
mu (6.5554212E+07,-5932777.)
Branch No.  0 (i*OMEGA)**2             6 =   (-9.2262792E+07,-1174482.)
iomega (-61.13672,9605.546)
imrio  (69.08185,69.52293)
imrioc (69.08194,69.52074)
(1.000000,0.0000000E+00)
(-0.2847293,-1.0817482E-04)
(-0.9189860,7.0201601E-08)
(0.5463914,2.0737553E-04)
(0.7635213,-4.6801066E-08)
(-0.7637883,-2.8993262E-04)
(-0.5462006,3.5100800E-08)
(0.9193074,3.4864456E-04)
(0.2846298,-5.2651203E-08)
(-1.000350,-3.7939285E-04)
mu (1.0224027E+08,-5.3946252E+07)
Branch No.  0 (i*OMEGA)**2             7 =   (-1.3377890E+08,-1480091.)
iomega (-63.98242,11566.46)
imrio  (75.83751,76.25818)
```

```
imrioc (75.83728,76.25863)
(1.000000,0.0000000E+00)
(-0.8308192,3.1516585E-05)
(-0.3097337,-3.5127923E-05)
(1.088145,-3.1303196E-05)
(-0.5943240,7.8435820E-05)
(-0.5943553,-9.8959827E-06)
(1.088124,-9.4306735E-05)
(-0.3096964,7.1078292E-05)
(-0.8308083,6.4218679E-05)
(0.9999602,-1.1546796E-04)
mu (1.4529238E+08,51596.13)
Branch No.  0 (i*OMEGA)**2          8 =  (-1.8406491E+08,-1867021.)
iomega (-68.80859,13567.23)
imrio  (82.15410,82.57182)
imrioc (82.15426,82.57207)
(1.000000,0.0000000E+00)
(-1.309705,4.0994073E-06)
(0.7153350,-9.3700746E-06)
(0.3728082,9.1626634E-06)
(-1.203591,-4.8802468E-07)
(1.203552,-1.0199717E-05)
(-0.3727369,1.2444630E-05)
(-0.7153528,-3.6601853E-06)
(1.309645,-7.3203705E-06)
(-0.9999297,8.7844446E-06)
mu (2.2543661E+08,-1.0230355E+07)
Branch No.  0 (i*OMEGA)**2          9 =  (-2.3761042E+08,-2284970.)
iomega (-74.11768,15414.79)
imrio  (87.58098,88.00310)
imrioc (87.58107,88.00335)
(1.000000,0.0000000E+00)
(-1.682495,5.6159638E-06)
(1.830793,-1.7751610E-05)
(-1.397818,2.7757062E-05)
(0.5210506,-2.6982447E-05)
(0.5211318,1.0650966E-05)
(-1.397843,1.6589685E-05)
(1.830745,-4.0538220E-05)
(-1.682407,4.7638863E-05)
(0.9999328,-3.1791518E-05)
mu (2.6666758E+08, 2.2426160E+07)
Branch No.  0 (i*OMEGA)**2         10 =  (-2.8153338E+08,-2638136.)
iomega (-78.61523,16779.14)
imrio  (91.38028,91.80943)
imrioc (91.38062,91.80936)
(1.000000,0.0000000E+00)
(-1.918977,-1.2857653E-05)
(2.682472,4.5136654E-05)
(-3.228632,-9.6926924E-05)
(3.513214,1.5842787E-04)
```

```
(-3.513170,-2.1399450E-04)
(3.228516,2.4510463E-04)
(-2.682320,-2.3890058E-04)
(1.918837,1.9061695E-04)
(-0.9999172,-1.0587333E-04)
Branch No.  1 (i*OMEGA)**2          1 =  (-1727935.,-313830.9)
iomega (118.8868,-1319.875)
imrio  (-26.87104,24.55943)
imrioc (-26.87065,24.55910)
(1.000000,0.0000000E+00)
(1.918982,-5.8519146E-07)
(2.682493,-2.1735682E-06)
(3.228678,-4.6815317E-06)
(3.513289,-7.5238904E-06)
(3.513272,-1.0115453E-05)
(3.228634,-1.1620231E-05)
(2.682436,-1.1411234E-05)
(1.918929,-9.0704680E-06)
(0.9999686,-5.0368267E-06)
mu (7951729.,327181.6)
Branch No.  1 (i*OMEGA)**2          2 =  (-7811573.,-500747.8)
iomega (89.53687,-2796.353)
imrio  (-37.99554,36.79843)
imrioc (-37.99553,36.79842)
(1.000000,0.0000000E+00)
(1.682507,8.6335419E-08)
(1.830829,8.6335419E-08)
(1.397876,5.1801248E-07)
(0.5211055,1.0791928E-06)
(-0.5211135,1.7698760E-06)
(-1.397884,2.3742239E-06)
(-1.830836,2.5037270E-06)
(-1.682512,2.0720499E-06)
(-1.000003,1.2518635E-06)
mu (1.4309340E+07,173221.7)
Branch No.  1 (i*OMEGA)**2          3 =  (-1.8618120E+07,-681858.1)
iomega (79.00098,-4315.595,
imrio  (-46.87920,46.02889)
imrioc (-46.87971,46.02890)
(1.000000,0.0000000E+00)
(1.309736,1.4812716E-05)
(0.7154124,4.4238837E-05)
(-0.3727181,7.0220878E-05)
(-1.203550,6.8387228E-05)
(-1.203589,2.7074264E-05)
(-0.3728245,-4.0890427E-05)
(0.7152732,-1.0104217E-04)
(1.309607,-1.1878873E-04)
(0.9999237,-7.9444944E-05)
mu (2.0918084E+07,-291307.8)
Branch No.  1 (i*OMEGA)**2          4 =  (-3.5003448E+07,-895241.5)
```

```
iomega (75.65332,-5916.855)
imrio  (-54.74026,54.04482)
imrioc (-54.73922,54.04443)
(1.000000,0.0000000E+00)
(0.8307768,-2.4223091E-05)
(-0.3098356,-5.1601903E-05)
(-1.088236,-3.5794073E-05)
(-0.5942835,3.0739073E-05)
(0.5945513,9.0317939E-05)
(1.088319,7.3224444E-05)
(0.3096766,-2.0351486E-05)
(-0.8310684,-1.0741143E-04)
(-1.000223,-1.0039872E-04)
mu (3.7189392E+07,-7475835.)
Branch No.  1 (i*OMEGA)**2                    5 =  (-5.8185504E+07,-1163387.)
iomega (76.25464,-7628.323)
imrio  (-62.06834,61.45099)
imrioc (-62.06834,61.45099)
(1.000000,0.0000000E+00)
(0.2846289,5.6180619E-07)
(-0.9189868,5.7375951E-07)
(-0.5461997,-5.7375951E-07)
(0.7635232,-1.3865855E-06)
(0.7635217,-1.6734653E-07)
(-0.5462028,1.6854185E-06)
(-0.9189881,1.2431456E-06)
(0.2846313,-1.1355658E-06)
(1.000003,-2.0320649E-06)
mu (8.2082576E+07,1550685.)
Branch No.  1 (i*OMEGA)**2                    6 =  (-8.9616080E+07,-1516512.)
iomega (80.09766,-9466.916)
imrio  (-69.09180,68.50970)
imrioc (-69.09160,68.50941)
(1.000000,0.0000000E+00)
(-0.2846528,4.2039628E-06)
(-0.9189866,9.3892947E-09)
(0.5462448,-8.0543250E-06)
(0.7635218,-1.5398443E-08)
(-0.7635833,1.1241990E-05)
(-0.5462007,1.4083942E-08)
(0.9190604,-1.3511946E-05)
(0.2846300,-9.2015089E-09)
(-1.000081,1.4693870E-05)
mu (1.0609301E+08,[illegible]96016.)
Branch No.  1 (i*OMEGA)**2                    7 =  (-1.3038785E+08,-1979836.)
iomega (86.69287,-11419.08)
imrio  (-75.84887,75.27522)
imrioc (-75.84859,75.27493)
(1.000000,0.0000000E+00)
(-0.8308573,2.5904540E-06)
(-0.3096915,-2.8543868E-06)
```

```
(1.088182,-2.5619736E-06)
(-0.5944169,6.3123384E-06)
(-0.5943430,-7.6089776E-07)
(1.088234,-7.5307648E-06)
(-0.3097799,5.6130239E-06)
(-0.8308829,5.1012153E-06)
(1.000095,-9.1055272E-06)
mu (1.5576734E+08,2273627.)
Branch No.  1 (i*OMEGA)**2            8 =  (-1.7975037E+08,-2559928.)
iomega (95.47070,-13407.44)
imrio  (-82.16828,81.58527)
imrioc (-82.16757,81.58463)
(1.000000,0.0000000E+00)
(-1.309778,-1.6930754E-06)
(0.7154927,3.6269312E-06)
(0.3727002,-2.5296204E-06)
(-1.203688,-2.1632156E-06)
(1.203830,6.3649736E-06)
(-0.3729604,-5.1696379E-06)
(-0.7154179,-1.4370679E-06)
(1.309977,7.5793437E-06)
(-1.000239,-7.0787491E-06)
mu (1.8128869E+08,2374305.)
Branch No.  1 (i*OMEGA)**2            9 =  (-2.3227365E+08,-3201124.)
iomega (105.0234,-15240.89)
imrio  (-87.59644,86.99490)
imrioc (-87.59653,86.99477)
(1.000000,0.0000000E+00)
(-1.682509,1.0381686E-05)
(1.830836,-2.9456172E-05)
(-1.397889,4.5897981E-05)
(0.5211259,-4.4134587E-05)
(0.5210884,1.7547038E-05)
(-1.397858,2.4339839E-05)
(1.830815,-6.1097962E-05)
(-1.682497,7.1529321E-05)
(0.9999952,-4.7711052E-05)
mu (2.7546778E+08,3563502.)
Branch No.  1 (i*OMEGA)**2           10 =  (-2.7534032[illegible]3749408.)
iomega (112.9785,-16593.77)
imrio  (-91.39783,90.77767)
imrioc (-91.39783,90.77767)
(1.000000,0.0000000E+00)
(-1.918984,0.0000000E+00)
(2.682505,-1.7627285E-07)
(-3.228705,3.5254570E-07)
(3.513334,-1.7627285E-07)
(-3.513334,1.7627285E-07)
(3.228705,-3.5254570E-07)
(-2.682505,-8.8136424E-08)
(1.918984,-8.8136424E-08)
```

```
(-0.9999995,8.8136424E-08)
Branch No.  2 (i*OMEGA)**2          1 =  (-1727934.,313829.3)
iomega (118.8856,1319.875)
imrio  (-26.87102,-24.55944)
imrioc (-26.87064,-24.55910)
(1.000000,0.0000000E+00)
(1.918982,4.8069217E-07)
(2.682494,1.7973708E-06)
(3.228679,4.1799321E-06)
(3.513290,6.7714900E-06)
(3.513273,9.1122520E-06)
(3.228635,1.0533428E-05)
(2.682437,1.0115436E-05)
(1.918930,8.1090684E-06)
(0.9999689,4.5143265E-06)
mu (7951728.,-327181.5)
Branch No.  2 (i*OMEGA)**2          2 =  (-7811571.,500747.9)
iomega (89.53564,2796.353)
imrio  (-37.99554,-36.79844)
imrioc (-37.99553,-36.79843)
(1.000000,0.0000000E+00)
(1.682507,-8.6335611E-08)
(1.830830,-1.7267122E-07)
(1.397876,-4.7484585E-07)
(0.5211055,-1.1439469E-06)
(-0.5211133,-1.7698800E-06)
(-1.397884,-2.2015581E-06)
(-1.830836,-2.5037327E-06)
(-1.682512,-1.9857191E-06)
(-1.000003,-1.2086986E-06)
mu (1.4309341E+07,-173222.6)
Branch No.  2 (i*OMEGA)**2          3 =  (-1.8618124E+07,681858.1)
iomega (78.99731,4315.595)
imrio  (-46.87918,-46.02891)
imrioc (-46.87973,-46.02890)
(1.000000,0.0000000E+00)
(1.309736,-1.4541678E-05)
(0.7154123,-4.4214990E-05)
(-0.3727185,-7.0240931E-05)
(-1.203551,-6.8690293E-05)
(-1.203590,-2.7265645E-05)
(-0.3728243,4.0643350E-05)
(0.7152737,1.0113801E-04)
(1.309608,1.1902810E-04)
(0.9999242,7.9644386E-05)
mu (2.0918078E+07,291304.7)
Branch No.  2 (i*OMEGA)**2          4 =  (-3.5003444E+07,895240.0)
iomega (75.64917,5916.854)
imrio  (-54.74024,-54.04484)
imrioc (-54.73922,-54.04443)
(1.000000,0.0000000E+00)
```

```
(0.8307768,2.4077115E-05)
(-0.3098354,5.1543728E-05)
(-1.088235,3.6144895E-05)
(-0.5942833,-3.0388594E-05)
(0.5945511,-9.0522946E-05)
(1.088319,-7.3604679E-05)
(0.3096764,2.0453861E-05)
(-0.8310683,1.0761653E-04)
(-1.000223,1.0048690E-04)
mu (3.7189316E+07,7475811.)
Branch No.  2 (i*OMEGA)**2          5 =  (-5.8185508E+07,1163387.)
iomega (76.25220,7628.323)
imrio  (-62.06833,-61.45100)
imrioc (-62.06834,-61.45098)
(1.000000,0.0000000E+00)
(0.2846288,-5.6155409E-07)
(-0.9189869,-5.9739796E-07)
(-0.5461996,5.2571016E-07)
(0.7635231,1.4098591E-06)
(0.7635214,1.6727142E-07)
(-0.5462027,-1.6488183E-06)
(-0.9189876,-1.2425877E-06)
(0.2846313,1.1350561E-06)
(1.000003,2.0789448E-06)
mu (8.2082536E+07,-1550466.)
Branch No.  2 (i*OMEGA)**2          6 =  (-8.9616080E+07,1516513.)
iomega (80.09375,9466.916)
imrio  (-69.09178,-68.50971)
imrioc (-69.09158,-68.50941)
(1.000000,0.0000000E+00)
(-0.2846528,-4.1972467E-06)
(-0.9189857,-5.6336358E-09)
(0.5462447,8.0410764E-06)
(0.7635210,1.0516120E-08)
(-0.7635829,-1.1221076E-05)
(-0.5462001,-1.2206211E-08)
(0.9190602,1.3489178E-05)
(0.2846296,7.9809839E-09)
(-1.000081,-1.4666983E-05)
mu (1.0609292E+08,-1785577.)
Branch No.  2 (i OMEGA)**2          7 =  (-1.3038786E+08,1979838.)
iomega (86.68945,11419.08)
imrio  (-75.84885,-75.27522)
imrioc (-75.84859,-75.27491)
(1.000000,0.0000000E+00)
(-0.8308578,-2.5664494E-06)
(-0.3096915,2.8234983E-06)
(1.088183,2.5298611E-06)
(-0.5944177,-6.2406507E-06)
(-0.5943429,7.4710334E-07)
(1.088235,7.4499058E-06)
```

```
(-0.3097804,-5.5536079E-06)
(-0.8308839,-5.0377544E-06)
(1.000096,9.0010326E-06)
mu (1.5576696E+08,-2271861.)
Branch No.  2 (i*OMEGA)**2          8 =  (-1.7973891E+08,2559314.)
iomega (95.44336,13407.01)
imrio  (-82.16689,-81.58404)
imrioc (-82.16759,-81.58463)
(1.000000,0.0000000E+00)
(-1.309667,-2.9475502E-06)
(0.7152544,6.2877903E-06)
(0.3728666,-4.3809155E-06)
(-1.203547,-3.7253233E-06)
(1.203413,1.1025740E-05)
(-0.3726203,-8.9557989E-06)
(-0.7153248,-2.4854505E-06)
(1.309479,1.3128959E-05)
(-0.9997736,-1.2258956E-05)
mu (1.8128790E+08,-2370810.)
Branch No.  2 (i*OMEGA)**2          9 =  (-2.3227318E+08,3202567.)
iomega (105.0605,15240.87)
imrio  (-87.59650,-86.99475)
imrioc (-87.59652,-86.99474)
(1.000000,0.0000000E+00)
(-1.682505,4.0948004E-07)
(1.830825,-1.0237001E-06)
(-1.397870,2.2521401E-06)
(0.5211027,-2.9175453E-06)
(0.5211092,3.0838964E-06)
(-1.397870,-2.6616203E-06)
(1.830817,2.0474001E-06)
(-1.682492,-1.0237001E-06)
(0.9999904,3.3270254E-07)
mu (2.7546762E+08,-3569768.)
Branch No.  2 (i*OMEGA)**2         10 =  (-2.7534032E+08,3749409.)
iomega (112.9727,16593.77)
imrio  (-91.39782,-90.77769)
imrioc (-91.39783,-90.77768)
(1.000000,0.0000000E+00)
(-1.918986,1.6818530E-07)
(2.682508,-3.3637059E-07)
(-3.228708,0.0000000E+00)
(3.513338,-5.0455589E-07)
(-3.513337,5.0455589E-07)
(3.228707,-3.3637059E-07)
(-2.682507,-8.4092648E-08)
(1.918986,-1.6818530E-07)
(-1.000000,4.2046324E-08)
Branch No.  3 (i*OMEGA)**2          1 =  (-2380412.,330272.2)
iomega (-106.7770,-1546.549)
imrio  (26.86497,-28.78374)
```

```
imrioc (26.86459,-28.78333)
(1.000000,0.0000000E+00)
(1.918982,5.0876679E-07)
(2.682489,2.0742032E-06)
(3.228670,4.3049499E-06)
(3.513276,7.0835990E-06)
(3.513254,9.5491614E-06)
(3.228611,1.0958054E-05)
(2.682414,1.0762375E-05)
(1.918912,8.5316278E-06)
(0.9999588,4.7354447E-06)
mu (8630993.,-344225.3)
Branch No.  3 (i*OMEGA)**2              2 =  (-8769234.,455740.9)
iomega (-76.92480,-2962.289)
imrio  (37.98922,-38.98854)
imrioc (37.98923,-38.98853)
(1.000000,0.0000000E+00)
(1.682507,-5.4724012E-08)
(1.830830,-1.6417204E-07)
(1.397877,-3.2834407E-07)
(0.5211071,-5.4724012E-07)
(-0.5211105,-7.9349820E-07)
(-1.397880,-8.7558419E-07)
(-1.830833,-1.0397563E-06)
(-1.682509,-8.2086018E-07)
(-1.000001,-6.0196413E-07)
mu (1.4748488E+07,-250720.2)
Branch No.  3 (i*OMEGA)**2              3 =  (-1.9881206E+07,584559.2)
iomega (-65.54541,-4459.316)
imrio  (46.87351,-47.56755)
imrioc (46.87297,-47.56709)
(1.000000,0.0000000E+00)
(1.309693,-2.0784639E-06)
(0.7152843,-6.6387597E-06)
(-0.3729225,-1.0470752E-05)
(-1.203749,-1.0173829E-05)
(-1.203668,-3.9776533E-06)
(-0.3727070,6.1933742E-06)
(0.7155666,1.5095475E-05)
(1.309952,1.7776189E-05)
(1.000154,1.1893744E-05)
mu (2.0617300E+07,44967.56)
Branch No.  3 (i*OMEGA)**2              4 =  (-3.6620688E+07,734676.0)
iomega (-60.69604,-6051.807)
imrio  (54.73306,-55.28475)
imrioc (54.73180,-55.28397)
(1.000000,0.0000000E+00)
(0.8307523,-1.6280941E-05)
(-0.3098881,-3.5177462E-05)
(-1.088272,-2.4729126E-05)
(-0.5942523,2.0533007E-05)
```

```
(0.5946431,6.1738894E-05)
(1.088393,5.0521270E-05)
(0.3096555,-1.3707321E-05)
(-0.8311775,-7.3683848E-05)
(-1.000325,-6.8900270E-05)
mu (4.2753092E+07,1.9644208E+07)
Branch No.  3 (i*OMEGA)**2          5 =  (-6.0250036E+07,924952.8)
iomega (-59.58057,-7762.318)
imrio  (62.06032,-62.53850)
imrioc (62.05998,-62.53775)
(1.000000,0.0000000E+00)
(0.2845813,1.7937400E-05)
(-0.9190400,2.0275309E-05)
(-0.5461520,-1.7735856E-05)
(0.7636414,-4.4682267E-05)
(0.7635363,-5.7439988E-06)
(-0.5463451,5.3832355E-05)
(-0.9190950,4.0550618E-05)
(0.2847284,-3.6650745E-05)
(1.000177,-6.5904831E-05)
mu (6.5548396E+07,5934552.)
Branch No.  3 (i*OMEGA)**2          6 =  (-9.2263024E+07,1174750.)
iomega (-61.15039,-9605.560)
imrio  (69.08185,-69.52303)
imrioc (69.08195,-69.52075)
(1.000000,0.0000000E+00)
(-0.2847335,1.1281406E-04)
(-0.9189858,-4.6849692E-08)
(0.5463995,-2.1665639E-04)
(0.7635210,-4.6849692E-08)
(-0.7637995,3.0271927E-04)
(-0.5462003,-5.8562115E-08)
(0.9193211,-3.6411581E-04)
(0.2846296,-5.8562115E-09)
(-1.000365,3.9641865E-04)
mu (1.0219393E+08,5.3960408E+07)
Branch No.  3 (i*OMEGA)**2          7 =  (-1.3378158E+08,1480224.)
iomega (-63.98242,-11566.57)
imrio  (75.83788,-76.25855)
imrioc (75.83728,-76.25863)
(1.000000,0.0000000E+00)
(-0.8308539,-3.0083331E-05)
(-0.3096941,3.3471013E-05)
(1.088179,2.9818808E-05)
(-0.5944111,-7.4787495E-05)
(-0.5943434,9.1861166E-06)
(1.088228,9.0141773E-05)
(-0.3097761,-6.7681482E-05)
(-0.8308793,-6.1308914E-05)
(1.000089,1.1015525E-04)
mu (1.4517026E+08,1099228.)
```

```
Branch No.  3 (i*OMEGA)**2              8 =  (-1.8406677E+08,1869110.)
iomega (-68.87793,-13567.29)
imrio  (82.15410,-82.57224)
imrioc (82.15426,-82.57206)
(1.000000,0.0000000E+00)
(-1.309723,1.5334019E-05)
(0.7153719,-3.2981487E-05)
(0.3727844,2.4261562E-05)
(-1.203616,1.6757680E-05)
(1.203618,-5.4484706E-05)
(-0.3727874,4.6046545E-05)
(-0.7153703,9.4020970E-06)
(1.309723,-6.2225867E-05)
(-1.000001,5.8844667E-05)
mu (2.2444466E+08,1.9398466E+07)
Branch No.  3 (i*OMEGA)**2              9 =  (-2.3761622E+08,2288260.)
iomega (-74.21582,-15414.98)
imrio  (87.58124,-88.00391)
imrioc (87.58106,-88.00334)
(1.000000,0.0000000E+00)
(-1.682535,1.6905027E-05)
(1.830914,-5.0646915E-05)
(-1.398011,7.9685386E-05)
(0.5212390,-7.7606339E-05)
(0.5210567,3.0418823E-05)
(-1.397954,4.6352492E-05)
(1.831022,-1.1479059E-04)
(-1.682733,1.3503572E-04)
(1.000151,-9.0148780E-05)
mu (2.6468098E+08,4.0762000E+07)
Branch No.  3 (i*OMEGA)**2             10 =  (-2.8153389E+08,2631452.)
iomega (-78.41504,-16779.15)
imrio  (91.38085,-91.80891)
imrioc (91.38062,-91.80937)
(1.000000,0.0000000E+00)
(-1.918978,-2.3632456E-05)
(2.682482,8.4262319E-05)
(-3.228652,-1.8137335E-04)
(3.513247,2.9666763E-04)
(-3.513215,-4.0003093E-04)
(3.228567,4.5876790E-04)
(-2.682370,-4.4752529E-04)
(1.918876,3.5741221E-04)
(-0.9999390,-1.9849541E-04)
```

```
OUTPUT FOR NONZEROB
eps0  9.9999997E-05  eps1  0.1000000
eps2  0.1000000     eps3  0.1000000
rhoe   2710.000      rhov   1.000000
ae  6.2500000E-02
le  0.9090000
e  5.5160001E+10
e0   4750000.      e1   1843750.     b  1.0000000E-03
n           10 iq            1 im           2 alpha  0.5000000
imax1             40  imax2           140  imax3          200
Branch No. 0 iomega**2 No.           1 =  (9.9995052E+11,349673.8)
iomega (999975.3,0.1748412)
imrio  (-999.9876,-8.7421693E-05)
imrioc (-999.9875,-4.7891485E-10)
(1.000000,0.0000000E+00)
(1.828169,1.4601265E-07)
(2.368327,4.8209864E-07)
(2.587787,9.2231738E-07)
(2.527601,1.3094324E-06)
(2.267370,1.5085000E-06)
(1.887427,1.4690710E-06)
(1.446003,1.2263185E-06)
(0.9755177,8.5809245E-07)
(0.4909331,4.3601196E-07)
Branch No. 0 iomega**2 No.           2 =  (9.9994803E+11,349672.9)
iomega (999974.0,0.1748410)
imrio  (-999.9870,-8.7421642E-05)
imrioc (-999.9870,-1.0777969E-10)
(1.000000,0.0000000E+00)
(1.515036,2.8566211E-07)
(1.336279,7.9574676E-07)
(0.6139306,1.1520580E-06)
(-0.2827702,1.0117719E-06)
(-0.9943206,3.6644997E-07)
(-1.328689,-4.6404875E-07)
(-1.282883,-1.0658343E-06)
(-0.9670860,-1.1677936E-06)
(-0.5094714,-7.4615866E-07)
Branch No. 0 iomega**2 No.           3 -  (9.9994488E+11,349671.8)
iomega (999972.4,0.1748407)
imrio  (-999.9862,-8.7421569E-05)
imrioc (-999.9865,5.7382971E-10)
(1.000000,0.0000000E+00)
(1.107533,4.2891546E-07)
(0.2390907,9.4432420E-07)
(-0.8652738,8.4621962E-07)
(-1.340219,3.7002444E-08)
(-0.8690234,-8.2836880E-07)
(0.1927339,-1.0929781E-06)
(1.165972,-7.2575108E-07)
(1.494040,-1.9971225E-07)
```

```
(1.015863,5.2901523E-08)
Branch No. 0 iomega**2 No.          4 =   (9.9993826E+11,349669.5)
iomega (999969.1,0.1748402)
imrio  (-999.9846,-8.7421424E-05)
imrioc (-999.9847,0.0000000E+00)
(1.000000,0.0000000E+00)
(0.6183243,3.4310446E-07)
(-0.5682581,3.5707581E-07)
(-0.9656028,-2.8586317E-07)
(-0.1499314,-7.0491149E-07)
(0.7980464,-2.4417554E-07)
(0.8046404,4.7162501E-07)
(-4.2990297E-02,5.3824533E-07)
(-0.8137011,3.2590012E-08)
(-0.7720997,-2.5448307E-07)
Branch No. 0 iomega**2 No.          5 =   (9.9993164E+11,349667.2)
iomega (999965.8,0.1748396)
imrio  (-999.9829,-8.7421286E-05)
imrioc (-999.9831,3.4177686E-10)
(1.000000,0.0000000E+00)
(5.6983080E-02,3.7243248E-07)
(-1.029749,1.0823873E-07)
(-0.2105127,-5.8872979E-07)
(1.038220,-3.2288793E-07)
(0.5065960,5.2806308E-07)
(-0.9208584,5.3225568E-07)
(-0.8921357,-1.5920646E-07)
(0.5672449,-4.5800851E-07)
(1.183969,-2.4459450E-07)
Branch No. 0 iomega**2 No.          6 =   (9.9992011E+11,349663.2)
iomega (999960.1,0.1748386)
imrio  (-999.9800,-8.7421031E-05)
imrioc (-999.9807,4.6639959E-10)
(1.000000,0.0000000E+00)
(-0.4683411,2.5848396E-07)
(-0.7823696,-2.2529204E-07)
(0.7759746,-2.9482896E-07)
(0.4966423,4.0715923E-07)
(-0.9131734,1.8424829E-07)
(-0.2615826,-4.0477300E-07)
(0.9499763,-6.2990537E-08)
(0.1055817,2.4263247E-07)
(-0.9506435,-8.7379393E-09)
Branch No. 0 iomega**2 No.          7 =   (9.9990752E+11,349658.8)
iomega (999953.8,0.1748375)
imrio  (-999.9769,-8.7420754E-05)
imrioc (-999.9769,5.2259147E-10)
(1.000000,0.0000000E+00)
(-1.011854,2.3266050E-07)
(-2.2162143E-02,-3.9487853E-07)
(1.064521,8.2989544E-08)
```

```
(-0.9396884,4.2366506E-07)
(-0.3040805,-4.1149025E-07)
(1.222867,-1.4429263E-07)
(-0.6190788,4.1436226E-07)
(-0.8099154,-2.5513055E-08)
(1.200566,-2.9474484E-07)
Branch No. 0 iomega**2 No.          8 =  (9.9989186E+11,349653.3)
iomega (999945.9,0.1748361)
imrio  (-999.9730,-8.7420412E-05)
imrioc (-999.9731,-8.9928703E-11)
(1.000000,0.0000000E+00)
(-1.435261,1.3215839E-07)
(1.029387,-3.4303750E-07)
(1.7084423E-02,4.1054108E-07)
(-1.067521,-1.8328214E-07)
(1.398843,-1.9796071E-07)
(-0.7235447,3.9354970E-07)
(-0.5042318,-2.3244714E-07)
(1.350638,-7.5578029E-08)
(-1.129440,1.7850256E-07)
Branch No. 0 iomega**2 No.          9 =  (9.9987548E+11,349647.6)
iomega (999937.8,0.1748347)
imrio  (-999.9689,-8.7420056E-05)
imrioc (-999.9688,2.0144153E-10)
(1.000000,0.0000000E+00)
(-1.795254,1.1206640E-07)
(2.182214,-3.5237954E-07)
(-2.000948,6.0828222E-07)
(1.220707,-7.0113902E-07)
(-1.8065292E-02,5.1479378E-07)
(-1.225691,-9.9155407E-08)
(2.045312,-3.3606989E-07)
(-2.091733,5.4383224E-07)
(1.311292,-4.0101364E-07)
Branch No. 0 iomega**2 No.         10 =  (9.9986250E+11,349643.0)
iomega (999931.3,0.1748335)
imrio  (-999.9656,-8.7419772E-05)
imrioc (-999.9655,1.2488019E-10)
(1.000000,0.0000000E+00)
(-2.007572,6.4654699E-08)
(3.005106,-2.3995383E-07)
(-3.928426,5.3225557E-07)
(4.667979,-8.9773010E-07)
(-5.087037,1.2520931E-06)
(5.053510,-1.4845094E-06)
(-4.481068,1.4920497E-06)
(3.366284,-1.2198240E-06)
(-1.808235,6.8745641E-07)
```

Vita

Captain Michele Lynn Devereaux [redacted]. She graduated from Vacaville High School in Vacaville, California in June, 1979. She attended the Massachusetts Institute of Technology on a four-year Air Force Reserve Officer Training Corps scholarship. Upon graduation in May 1983, she was commissioned a second lieutenant in the United States Air Force. Captain Devereaux's first assignment on active duty was to Palmdale Plant 42. She supported the test and checkout of the fourth shuttle orbiter, OV-104 Atlantis, and the modification of the original orbiter, OV-102 Columbia. When Atlantis was completed, she was transferred to Vandenberg AFB. As part of the 6595th Shuttle Test Group, she helped prepare the facilities for a Shuttle launch. In May, 1987 she was assigned to the Air Force Institute of Technology, Wright- Patterson Air Force Base, Ohio.

[redacted]

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GAE/AA88D-08	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL *(If applicable)* AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State, and ZIP Code)* Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS *(City, State, and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)*	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

12 Jan 1989

11. TITLE *(Include Security Classification)*
IMPROVED SOLUTION TECHNIQUES FOR THE EIGENSTRUCTURE OF FRACTIONAL ORDER SYSTEMS

12. PERSONAL AUTHOR(S)
Michele Lynn Devereaux, B.S., Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1988 December	96

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Viscoelasticity, Numerical Methods, Damping
20	11		
13	13		

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Thesis Advisor: Ronald H. Bagley, LtCol, USAF
Associate Professor
Department of Aeronautics and Astronautics

The structural problem of a viscoelastically damped rod is considered. A four parameter fractional derivative viscoelastic model rather than the traditional viscous model is used to describe the stress-strain relationship. The introduction of fractional order derivatives leads to high order matrix equations, which are cumbersome and time consuming to solve. Thus, there exists a motivation to seek alternate solution techniques. An existing technique, modified matrix iteration, is presented, and a new one, employing spectrum shift concepts, is proposed. The spectrum shift technique is shown to be significantly more efficient.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Ronald H. Bagley, LtCol, USAF	22b. TELEPHONE *(Include Area Code)* (513) 255-3517 22c. OFFICE SYMBOL ENY